INTERNATIONAL STANDARD

ISO/IEC 14776-322

First edition
2007-02

Information technology – Small computer system interface (SCSI) –

Part 322: Block commands-2 (SBC-2)

Reference number
ISO/IEC 14776-322:2007(E)

INTERNATIONAL STANDARD

ISO/IEC 14776-322

First edition
2007-02

Information technology – Small computer system interface (SCSI) –

Part 322: Block commands-2 (SBC-2)

International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland
Telephone: +41 22 919 02 11 Telefax: +41 22 919 03 00 E-mail: inmail@iec.ch Web: www.iec.ch

PRICE CODE XB

For price, see current catalogue

CONTENTS

INFORMATION TECHNOLOGY – SMALL COMPUTER SYSTEM INTERFACE –

Part 322: Block commands-2 (SBC-2)

FOREWORD

1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards. Their preparation is entrusted to technical committees; any ISO and IEC member body interested in the subject dealt with may participate in this preparatory work. International governmental and non-governmental organizations liaising with ISO and IEC also participate in this preparation.

2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

3) The formal decisions or agreements of IEC and ISO on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC and ISO member bodies.

4) IEC, ISO and ISO/IEC Publications have the form of recommendations for international use and are accepted by IEC and ISO member bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC, ISO and ISO/IEC Publications is accurate, IEC or ISO cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.

5) In order to promote international uniformity, IEC and ISO member bodies undertake to apply IEC, ISO and ISO/IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any ISO/IEC Publication and the corresponding national or regional publication should be clearly indicated in the latter.

6) ISO and IEC provide no marking procedure to indicate their approval and cannot be rendered responsible for any equipment declared to be in conformity with an ISO/IEC Publication.

7) All users should ensure that they have the latest edition of this publication.

8) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC or ISO member bodies for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon, this ISO/IEC publication or any other IEC, ISO or ISO/IEC publications.

9) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.

10) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

International Standard ISO/IEC 14776-322 was prepared by subcommittee 25: Interconnection of Information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

The list of all currently available parts of the ISO/IEC 14776 series, under the general title *Information technology – Small computer system interface,* can be found on the IEC web site.

This International Standard has been approved by vote of the member bodies and the voting results may be obtained from the address given on the title page.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

INTRODUCTION

The set of SCSI standards specifies the interfaces, functions and operations necessary to ensure interoperability between conforming SCSI implementations. ISO/IEC 14776-322 (this standard) describes the functions. Conforming implementations may employ any design technique that does not violate interoperability.

This standard makes the following concepts from previous standards obsolete:

- optical-memory device type, model, commands (the ERASE, MEDIUM SCAN, READ GENERATION, READ UPDATED BLOCK and UPDATE BLOCK commands) and parameters (Optical-Memory mode page);
- write-once device type, model, commands and parameters;
- extent reservations and RESERVE/RELEASE reservations;
- sequential media model;
- rotational position locking model;
- relative addressing (including the RELADR bit in many CDBs) and the SET LIMITS commands;
- CHANGE DEFINITION, COMPARE, COPY, COPY AND VERIFY, LOCK UNLOCK CACHE, RESERVE, RELEASE, REZERO UNIT, SEEK, SEARCH DATA HIGH, SEARCH DATA EQUAL and SEARCH DATA LOW commands;
- third-party XOR operation and hybrid XOR operation model, commands (REBUILD, REGENERATE and XDWRITE EXTENDED commands) and mode page fields (XOR Control mode page MAXIMUM REGENERATE SIZE field, MAXIMUM REBUILD TRANSFER SIZE field and REBUILD DELAY field);
- the following mode pages and mode page fields:
  * Caching mode page NON CACHE SEGMENT SIZE field:
  * Flexible Disk mode page;
  * Format Device mode page;
  * Medium Types Supported mode page and all medium types in the mode parameter header;
  * Notch and Partition mode page;
  * the following Read-Write Error Recovery mode page fields:
  * CORRECTION SPAN FIELD,
  * HEAD OFFSET COUNT field and
  * DATA STROBE OFFSET COUNT field;
  * Rigid Disk Geometry mode page and
  * Verify Error Recovery mode page VERIFY CORRECTION SPAN field;
- Device Status Output and Device Status Input diagnostic pages;
- DISABLE SAVING PARAMETERS (DSP) bit in the Format Unit parameter list;
- INTERERLEAVE FIELD in the format unit command and
- erase by-pass (EBP) bit in the WRITE and WRITE AND VERIFY commands. This bit was formerly reserved for the direct-access block device type, so is just marked reserved in this standard.

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

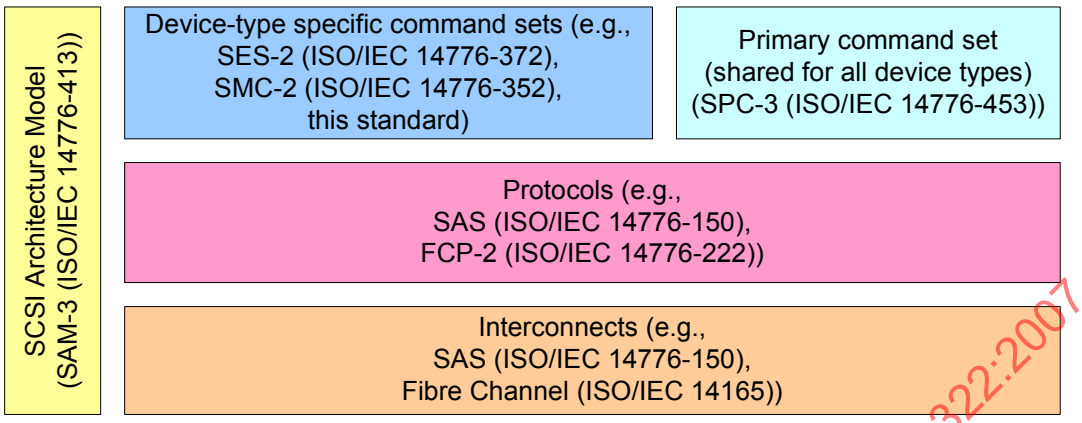


Figure 1 — SCSI document relationships

Figure 1 shows the general relationship of SCSI standards and does not imply a relationship with respect to hierarchy, protocol stack or system architecture.

The standard is organized as follows:

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

Informative Annex A (Numeric order codes) summarizes service action assignments for variable-length commands and commands using the SERVICE ACTION IN and SERVICE ACTION OUT operation codes.
Informative Annex B (XOR command examples) provides examples of XOR command usage.
Informative Annex C (CRC example in C) provides example C code for the protection information CRC.
Bibliography

INFORMATION TECHNOLOGY – SMALL COMPUTER SYSTEM INTERFACE –

Part 322: Block commands-2 (SBC-2)

1 Scope

This part of ISO/IEC 14776 defines the command set extensions to facilitate operation of SCSI direct-access block devices. The clauses of this standard, implemented in conjunction with the applicable clauses of SPC-3, fully specify the standard command set for SCSI direct-access block devices.

The objective of this standard is to

a) permit an application client to communicate over a SCSI service delivery subsystem with a logical unit that declares itself to be a direct-access block device in the PERIPHERAL DEVICE TYPE field of the standard INQUIRY data (see SPC-3) and
b) define commands unique to the direct-access block device type.

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

2 Normative references

The following referenced documents are indispensable for the application of this document. 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-3) – Part 342: Controller commands-2 (SCC-2)*

ISO/IEC 14776-352 (under consideration), *Information technology – Small computer system interface (SCSI) – Part 352: Media changer commands-2 (SMC-2) [ANSI INCITS 382-2004]*

ISO/IEC 14776-364 (under consideration), *Information technology – Small computer system interface (SCSI) – Part 364: Multimedia commands-4 (MMC-4) [ANSI INCITS 401-2005]*

ISO/IEC 14776-372 (under consideration), *Information technology – Small computer system interface (SCSI) – Part 372: Enclosure services-2 (SES-2) [INCITS working draft T10#1559-D]*

ISO/IEC 14776-413 (under consideration), *Information technology – Small computer system interface (SCSI) – Part 413: Architecture model-3 (SAM-3)*

ISO/IEC 14776-453 (under consideration), *Information technology – Small computer system interface (SCSI) – Part 453: Primary commands-3 (SPC-3) [ANSI INCITS 408-2005]*

IEC 60027:2000, *Letter symbols to be used in electrical technology – Part 2: Telecommunications and electronics*

3 Definitions, symbols, abbreviations, keywords and conventions

3.1 Definitions

For the purposes of this document the following definitions apply.

3.1.1
additional sense code: combination of the ADDITIONAL SENSE CODE and ADDITIONAL SENSE CODE QUALIFIER fields in the sense data (see SPC-3)

3.1.2
application client: an object that is the source of SCSI commands (see SAM-3)

3.1.3
byte: sequence of eight contiguous bits considered as a unit

3.1.4
cache: temporary and often volatile data storage area outside the area accessible by application clients that may contain a subset of the data stored in the non-volatile data storage area

3.1.5
check data: information contained within a redundancy group (see 3.1.37) that may allow lost or destroyed XOR-protected data (see 3.1.46) to be recreated

3.1.6
command: request describing a unit of work to be performed by a device server (see SAM-3)

3.1.7
command descriptor block (CDB): structure used to communicate commands from an application client to a device server (see SPC-3)

3.1.8
cyclic redundancy check (CRC): an error checking mechanism that checks data integrity by computing a polynomial algorithm based checksum (see 4.16.3)

3.1.9
data defect list (DLIST): list of defects sent by the application client to the device server during a FORM AT UNIT command (see 4.8)

3.1.10
data-in buffer: the buffer identified by the application client to receive data from the device server during the processing of a command (see SAM-3)

3.1.11
data-out buffer: the buffer identified by the application client to supply data that is sent from the application client to the device server during the processing of a command (see SAM-3)

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

3.1.13
device server: an object within a logical unit that processes SCSI tasks according to the rules of task management (see SAM-3)

3.1.14
device type: type of device (or device model) implemented by the device server as indicated by the PERIPHERAL DEVICE TYPE field of the standard INQUIRY data (see SPC-3)

3.1.15
direct-access block device: a device that is capable of containing data stored in blocks that each have a unique logical block address

3.1.16
domain: I/O system consisting of a set of SCSI devices that interact with one another by means of a service delivery subsystem (see SAM-3)

3.1.17
error correcting code (ECC): an error checking mechanism that checks data integrity and enables some errors in the data to be corrected

3.1.18
exclusive-or (XOR): boolean arithmetic function on two binary input values that results in an output value of 1 if one and only one of the input values is 1

3.1.19 extent: fixed set of logical blocks occupying contiguous logical block addresses on a single logical unit

3.1.20
field: group of one or more contiguous bits, a part of a larger structure such as a CDB (see 3.1.7) or sense data (see SPC-3)

3.1.21 format corrupt: vendor-specific condition in which the application client may not be able to perform read, write or verify operations (see 4.6)

3.1.22
grown defect list (GLIST): all the defects sent by the application client to the device server (see 4.8)

3.1.23
hard reset: condition resulting from the events defined by SAM-3 in which the SCSI device performs the hard reset operations described in SAM-3, this standard and other applicable command standards (see table 9 in 5.1)

3.1.24
I_T nexus loss: condition resulting from the events defined by SAM-3 in which the SCSI device performs the I_T nexus loss operations described in SAM-3, this standard and other applicable command standards (see table 9 in 5.1)

3.1.25
logical block: set of data bytes accessed and referenced as a unit

3.1.26
logical block address (LBA): value used to reference a logical block

3.1.27
logical unit certification list (CLIST): defects detected by the device server during an optional certification process performed during the FORMAT UNIT command (see 4.8)

3.1.28
logical unit reset: condition resulting from the events defined in SAM-3 in which the logical unit performs the logical unit reset operations described in SAM-3, this standard and other applicable command standards (see table 9 in 5.1)

3.1.29
media: plural of medium

3.1.30
medium: material on which data is stored (e.g., a magnetic disk)

3.1.31
non-volatile cache:
cache that retains data through power cycles

3.1.32
non-volatile medium: physical storage medium that retains data written to it for subsequent read operations through power cycles (e.g., a disk within a device that stores data as magnetic field changes that do not require device power to exist)

3.1.33
power cycle: power being removed followed by power being applied to a SCSI device

3.1.34
power on: condition resulting from the events defined by SAM-3 in which the SCSI device performs the power on operations described in SAM-3, this standard and other applicable command standards (see table 9 in 5.1)

3.1.35
primary defect list (PLIST): list of defects that are considered permanent defects (see 4.8)

3.1.36
protection information: fields appended to each logical block that contain a cyclic redundancy check (CRC), an application tag and a reference tag

3.1.37
redundancy group: a grouping of XOR-protected data (see 3.1.46) and associated check data (see 3.1.5) into a single type of data redundancy (see SCC-2)

NOTE This standard only supports the XOR (see 3.1.18) type of redundancy.

3.1.38
sense data: data describing an error or exceptional condition that a device server delivers to an application client in association with CHECK CONDITION status (see SPC-3)

3.1.39
sense key: contents of the SENSE KEY field in the sense data (see SPC-3)

3.1.40
status: one byte of response information sent from a device server to an application client upon completion of each command (see SAM-3)

3.1.41 storage array controller: any combination of an initiator and application clients (see SAM-3) that originates SCSI commands, converts input LUNs to output LUNs and converts input LBAs to output LBAs

NOTE A storage array controller organizes a group of direct-access block devices into various objects (e.g., redundancy groups and volume sets) (see SCC-2).

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

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

3.1.44
volatile medium: medium that does not retain data written to it for a subsequent read operation through power cycles (e.g., a silicon memory device that loses data written to it if device power is lost)

3.1.45
XOR operation: performing a XOR (see 3.1.18) bitwise on two identical-sized multiple-bit input values (e.g., the current value of a logical block and the new value for that logical block)

NOTE In a storage array implementing a redundancy group (see 3.1.37), the XOR operation is used in error correction algorithms and may be performed by the storage array controller (see 3.1.41) or by the direct-access block devices (see 3.1.15) and 4.14.

3.1.46
XOR-protected data: logical blocks, including user data and protection information, if any, that are part of a redundancy group (see 3.1.37)

3.2 Symbols and abbreviations

Symbols and abbreviations used in this standard:

Abbreviation	Meaning
CDB	command descriptor block (see 3.1.7)
CRC	cyclic redundancy check (see 3.1.8)
CLIST	logical unit certification list (see 3.1.27)
DLIST	data defect list (see 3.1.9)
ECC	error correcting code (see 3.1.17)
GLIST	grown defect list (see 3.1.22)
I/O	input/output
LBA	logical block address (see 3.1.26)
LSB	least significant bit
LUN	logical unit number
MMC-4	SCSI Multimedia Commands - 4 standard
MSB	most significant bit
PLIST	primary defect list (see 3.1.35)
SAM-3	SCSI Architecture Model - 3 standard
SCSI	Small Computer System Interface family of standards
SCC-2	SCSI-3 Controller Commands - 2 standard
SES-2	SCSI Enclosure Services - 2 standard
SMC-2	SCSI Media Changer Commands - 2 standard
SPC-3	SCSI Primary Commands - 3 standard
XOR	exclusive-or (see 3.1.18)

3.3 Keywords

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

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

3.3.3
invalid: A keyword used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt of an invalid bit, byte, word, field or code value shall be reported as an error.

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

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

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

3.3.7
need not: Keywords indicating a feature that is not required to be implemented (equivalent to “is not required to”).

3.3.8
obsolete: A keyword indicating that an item was defined in prior SCSI standards but has been removed from this standard.

3.3.9
optional: A 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.

3.3.10
reserved: A keyword referring to bits, bytes, words, fields and code values that are set aside for future standardization. A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard. Recipients are not required to check reserved bits, bytes, words or fields for zero values. Receipt of reserved code values in defined fields shall be reported as an error.

3.3.11
restricted: A keyword referring to bits, bytes, words and fields that are set aside for use in other SCSI standards. A restricted bit, byte, word or field shall be treated as a reserved bit, byte, word or field for the purposes of the requirements defined in this standard.

3.3.12
shall: A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard.

3.3.13
should: A keyword indicating flexibility of choice with a strongly preferred alternative; equivalent to the phrase “it is strongly recommended.”

3.3.14
vendor-specific: Something (e.g., a bit, field or code value) that is not defined by this standard and may be used differently in various implementations.

3.4 Conventions

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

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

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

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

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

A hexadecimal number is represented in this standard by any sequence of digits comprised of only the Western-Arabic numerals 0 through 9 and/or the upper-case English letters A through F immediately followed by a lower-case h (e.g., FA23h). Underscores or spaces may be included in hexadecimal number representations to increase readability or delineate field boundaries (e.g., B FD8CFA23h or B_FD8C_FA23h).

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

This standard uses the ISO convention for representing decimal numbers (e.g., the thousands and higher multiples are separated by a space and a comma is used as the decimal point). Table 1 shows some examples of decimal numbers represented using the ISO and American conventions.

Table 1 — ISO and American numbering convention examples

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

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

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

Notes do not constitute any requirements for implementers.

4 Direct-access block device type model

4.1 Direct-access block device type model overview

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

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

Direct-access block devices store data for later retrieval in logical blocks. Logical blocks contain user data, may contain protection information accessible to the application client and may contain additional information not normally accessible to the application client (e.g., an ECC). The number of bytes of user data contained in each logical block is the block length. The block length is greater than or equal to one byte and should be even. Most direct-access block devices support a block length of 512 bytes and some support additional block lengths (e.g., 520 or 4 096 bytes). The block length does not include the length of protection information and additional information, if any, that are associated with the logical block. The block length is the same for all logical blocks on the medium.

Each logical block is stored at a unique logical block address (LBA), which is either four bytes (i.e., a short LBA) or eight bytes (i.e., a long LBA) in length. The logical block addresses on a logical block shall begin with zero and shall be contiguous up to the last logical block on the logical unit. An application client uses commands performing write operations to store logical blocks and commands performing read operations to retrieve logical blocks. A write operation causes one or more logical blocks to be written to the medium. A read operation causes one or more logical blocks to be read from the medium. A verify operation confirms that one or more logical blocks were correctly written and are able to be read without error from the medium.

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). The medium may contain vendor specific information that is not addressable through an LBA. Such data may include defect management data and other device management information.

4.2 Media examples

4.2.1 Media examples overview

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

a) rotating media (see 4.2.2) and
b) memory media (see 4.2.3).

Other types of media are possible.

4.2.2 Rotating media

The typical application of a direct-access block device is a magnetic disk device. The medium is a spinning disk 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 is read by using the read portion of the head to read the recorded data.

The circular path followed by the read-write head at a particular radius is called a track. The track is divided into sectors each containing blocks of stored data. If there are more than one disk spinning on a single axis and the actuator has one or more read-write heads to access the disk surfaces, 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 media-based direct-access block device is ready when the disks are rotating at the correct speed and the read-write circuitry is powered and ready to access the data and may require a START STOP UNIT command (see 5.17) to bring the logical unit to the ready state.

Rotating media-based direct-access block device are usually non-volatile.

The defect management scheme of a disk device may not be discernible through this command set, though some aspects (see 4.8) may be accessible to the application client with the READ LONG commands and the WRITE LONG commands (see 5.14, 5.15, 5.33, and 5.34).

4.2.3 Memory media

Memory media is based on solid state random access memories (RAMs) (e.g., static RAM (SRAM), dynamic RAM (DRAM), magnetoresistive RAM (MRAM), ferroelectric RAM (FeRAM) or flash memory). Memory media-based direct-access block devices may be used for fast-access storage.

A memory media-based direct-access block device is ready after power on and does not require a START STOP UNIT command (see 5.17) to bring the logical unit to a ready state.

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

The defect management scheme (e.g., ECC bytes) (see 4.8) may be accessible to the application client with the READ LONG commands and the WRITE LONG commands (see 5.14, 5.15, 5.33 and 5.34).

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

4.3 Removable medium

4.3.1 Removable medium overview

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

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

An application client may check whether a removable medium is mounted by issuing a TEST UNIT READY command (see SPC-3). A direct-access block device containing a removable medium may not be accessible for write, read, and verify operations until it receives a START STOP UNIT command (see 5.17).

If the direct-access block device implements cache, either volatile or non-volatile, it ensures that all logical blocks of the medium contain the most recent user data and protection information, if any, prior to permitting unmounting of the removable medium.

The PREVENT ALLOW MEDIUM REMOVAL command (see SPC-3) allows an application client to restrict the unmounting of the removable medium. This is useful in maintaining system integrity.

If the application client issues a START STOP UNIT command to eject the removable medium and the direct-access block device is prevented from unmounting by the PREVENT ALLOW MEDIUM REMOVAL command, the START STOP UNIT command is rejected by the device server.

4.3.2 Removable medium with an attached media changer

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

The direct-access block device indicates its ability to support these commands by setting the MCHNGR bit to one in its standard INQUIRY data (see SPC-3). A MCHNGR bit set to one indicates that the MOVE MEDIUM ATTACHED and READ ELEMENT STATUS ATTACHED commands (see SMC-2) are supported. The logical unit may require a MODE MEDIUM ATTACHED command (see SMC-2) to become ready.

4.4 Logical blocks

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

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

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

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

NOTE If a command with a 4-byte LOGICAL BLOCK ADDRESS field accesses logical blocks beyond logical block address FFFFFFFFh and fixed format sense data is used, there is no field in the sense data large enough to report the logical block address of an error (see 4.13).

If a command is received that references or attempts to access a logical block not within the capacity of the medium, the device server terminates 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 command may be terminated before processing or after the device server has transferred some or all of the data.

The number of bytes of user data contained in a logical block is the block length. The parameter data returned by the READ CAPACITY command (see 5.10) describes the block length that is used on the medium. The mode parameter block descriptor (see 6.3.2) is used to change the block length in direct-access block devices that support changeable block lengths. The block length does not include the length of protection information and additional information, if any.

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

4.5 Ready state

A direct-access block device is ready when the device server is capable of processing medium access commands (i.e., commands that perform read operations, write operations or verify operations).

A direct-access block device using removable media is not ready until a volume is mounted and other conditions are met (see 4.2). A direct-access block device that is not ready shall terminate medium 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 5.17). An application client may need to issue a START STOP UNIT command with a START bit set to one to make a direct-access block device ready.

4.6 Initialization

Direct-access block devices may require initialization prior to write, read and verify operations. This initialization is performed by a FORMAT UNIT command (see 5.2). Parameters related to the format (e.g., block size) may be set with the MODE SELECT command prior to the format operation. Some direct-access block devices are initialized by means not specified in this standard. The time when the initialization occurs is vendor-specific.

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

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

NOTE 1 Mode parameter block descriptors read with the MODE SENSE command before a FORMAT UNIT completes return information that may not reflect the true state of the medium.

A direct-access block device may become format corrupt after processing a MODE SELECT command that changes parameters related to the medium format. During this time, the device server may terminate medium 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 parameter data returned by the READ CAPACITY (10) command (see 5.10) or the READ CAPACITY (16) command (see 5.11) changes (e.g., when a FORMAT UNIT command or a MODE SELECT command completes changing the number of blocks, block size, protection information or reference tag ownership values, or when a vendor-specific mechanism causes a change), the device server should establish a unit attention condition for the initiator port associated with each I_T nexus except the I_T nexus on which the command causing the change was received with an additional sense code of CAPACITY DATA HAS CHANGED.

NOTE 2 Logical units compliant with previous versions of this standard did not establish a unit attention condition.

4.7 Write protection

Write protection prevents the alteration of the medium by commands issued to the device server. Write protection is usually controlled by the user of the medium through manual intervention (e.g., mechanical lock) or may result from hardware controls (e.g., tabs on the media housing) or software write protection. All sources of write protection are independent. When present, any write protection shall cause otherwise valid commands that request alteration of the medium to be rejected with CHECK CONDITION status with the sense key set to DATA PROTECT. Only when all write protections are disabled shall the device server process commands that request alteration of the medium.

Hardware write protection results when a physical attribute of the drive or medium is changed to specify that writing shall be prohibited. Changing the state of the hardware write protection requires physical intervention, either with the drive or the medium. If allowed by the drive, 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-3). Software write protection is optional. Changing the state of software write protection shall not prevent previously accepted data (e.g., data in cache) from being written to the media.

The device server reports the status of write protection in the device server and on the medium with the DEVICE-SPECIFIC PARAMETER field in the mode parameter header (see 6.3.1).

4.8 Medium defects

Any medium has the potential for defects that cause data to be lost. Therefore, each logical block may contain additional information that allows the detection of changes to the user data and protection information, if any, caused by defects in the medium or other phenomena and may also allow the data to be reconstructed

following the detection of such a change (e.g., ECC bytes). Some direct-access block devices allow the application client to examine and modify the additional information by using the READ LONG commands and the WRITE LONG commands (see 5.14, 5.15, 5.33, and 5.34). The application client may use the WRITE LONG commands to induce a defect to test the defect detection logic of the direct-access block device or to emulate an unrecoverable logical block when generating a mirror copy.

Defects may also be detected and managed during processing of the FORMAT UNIT command (see 5.2). The FORMAT UNIT command defines four sources of defect information: the PLIST, CLIST, DLIST and GLIST. These defects may be reassigned or avoided during the initialization process so that they do not affect any logical blocks. The sources of defect location information (i.e., defects) are defined as follows.

a) Primary defect list (PLIST). This is the list of defects, which may be supplied by the original manufacturer of the device or medium, that are considered permanent defects. The PLIST is located outside of the application client accessible logical block space. The PLIST is accessible by the device server for reference during the format operation, but it is not accessible by the application client except through the READ DEFECT DATA commands (see 5.10 and 5.13). Once created, the original PLIST shall not change.
b) Logical unit certification list (CLIST). This list includes defects detected by the device server during an optional certification process performed during the FORMAT UNIT command. This list shall be added to the GLIST.
c) Data defect list (DLIST). This list of defects may be supplied by the application client to the device server during the FORMAT UNIT command. This list shall be added to the GLIST.
d) Grown defect list (GLIST). The GLIST includes all defects sent by the application client (i.e., the DLIST) or detected by the device server (i.e., the CLIST). The GLIST does not include the PLIST. If the CMPLST bit is set to zero, the GLIST shall include DLISTs provided to the device server during the previous and the current FORMAT UNIT commands. The GLIST shall also include
   A) defects detected by the format operation during medium certification,
   B) defects previously identified with a REASSIGN BLOCKS command (see 5.16) and
   C) defects previously detected by the device server and automatically reallocated.

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

Defects may also occur after initialization. The application client issues a REASSIGN BLOCKS command (see 5.16) to request that the specified logical block address be reassigned to a different part of the medium. This operation may be repeated if a new defect appears at a later time. The total number of defects that may be handled in this manner is vendor-specific.

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

4.9 Write failures

If one or more commands performing write operations are in the task set and are being processed when power is lost (e.g., resulting in a vendor-specific command timeout by the application client) or a medium error or hardware error occurs (e.g., because a removable medium was incorrectly unmounted), the data in the logical blocks being written by those commands is indeterminate. When accessed by a command performing a read or verify operation (e.g., after power on or after the removable medium is mounted), the device server may return old data, new data, or vendor-specific data in those logical blocks.

Before reading logical blocks which encountered such a failure, an application client should reissue any commands performing write operations that were outstanding.

4.10 Caches

Direct-access block devices may implement caches. A cache is an area of temporary storage in the direct-access block device with a fast access time that is used to enhance performance. Cache exists separately from the medium and is not directly accessible by the application client. Use of cache for write or read operations may reduce the access time to a logical block and increase the overall data throughput.

Cache stores user data and protection information, if any.

Cache may be volatile or non-volatile. Volatile caches do not retain data through power cycles. Non-volatile cache memories retain 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.

During read operations, the device server uses 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 to advise the device server about future requests or to restrict the use of cache for a particular request.

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

When the cache becomes full of logical blocks, new logical blocks may replace those currently in the cache. The disable page out (DPO) bit in the CDB of commands performing write, read or verify operations allows the application client to influence the replacement of logical blocks in the cache. For write operations, setting the DPO bit to one specifies that the device server should not replace existing logical blocks in the cache with the new logical blocks being written. For read and verify operations, setting the DPO bit to one specifies that the device server should not replace logical blocks in the cache with the logical blocks that are being read.

NOTE This does not mean that stale data is allowed in the cache. If a write operation accesses the same LBA as a logical block in the cache, the logical block in the cache is updated with the new write data.

Application clients may use the force unit access (FUA) bit in the CDB of commands performing write or read operations to specify that the device server shall access the medium. For a write operation, setting the FUA bit to one causes the device server to complete the data write to the medium before completing the command. For a read operation, setting the FUA bit to one causes the device server to retrieve the logical blocks from the medium rather than from the cache.

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

Application clients may use the force unit access non-volatile cache (FUA_NV) bit in the CDB of commands performing write or read operations to specify that the device server may access a non-volatile cache, if any, rather than the medium, if the FUA bit is set to zero. For a write operation, an FUA_NV bit set to one with the FUA bit set to zero allows the device server to complete the data write to non-volatile cache rather than the medium before completing the command. For a read operation, an FUA_NV bit set to one with the FUA bit set to zero allows the device server to retrieve the logical blocks from the non-volatile cache rather than the medium.

When a VERIFY command or a WRITE AND VERIFY command is processed, both a force unit access and a synchronize cache operation are implied, since the logical blocks are being verified as being stored on the medium. The DPO bit is defined in the VERIFY command since the VERIFY command may cause the replacement of logical blocks in the cache.

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

a) the PRE-FETCH commands (see 5.3 and 5.4) cause a set of logical blocks requested by the application client to be read into cache for possible future access. The logical blocks fetched are subject to later replacement;
b) the SYNCHRONIZE CACHE commands (see 5.18 and 5.19) force any write data in cache in the requested set of logical blocks to be written to the medium. These commands may be used to ensure that the data is written and any detected errors reported;
c) the Caching mode page (see 6.3.3), writable by the MODE SELECT commands, allows control of cache behavior.

4.11 Implicit HEAD OF QUEUE command processing

Each of the following commands defined in this standard may be processed by the task manager as if it has a task attribute of HEAD OF QUEUE (see SAM-3) if it is received with a SIMPLE task attribute, an ORDERED task attribute, or no task attribute:

a) the READ CAPACITY (10) command and
b) the READ CAPACITY (16) command.

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

Application clients should not send a command with the ORDERED task attribute if it may be processed as if it has a task attribute of HEAD OF QUEUE, because whether the ORDERED task attribute is honored is vendor-specific.

4.12 Reservations

Reservation restrictions are placed on commands as a result of access qualifiers associated with the type of reservation. See SPC-3 for a description of reservations. The details of commands that are allowed under what types of reservations are described in table 2.

Commands from I_T nexuses holding a reservation should complete normally. Table 2 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-3 defines the conditions that result in RESERVATION CONFLICT.

Table 2 — SBC-2 commands that are allowed in the presence of various reservations

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
FORMAT UNIT	Conflict	Conflict	Allowed	Conflict	Conflict
PRE-FETCH (10)/(16)	Allowed	Conflict	Allowed	Allowed	Conflict
READ (6)/(10)/(12)/(16)/(32)	Allowed	Conflict	Allowed	Allowed	Conflict
READ CAPACITY (10)/(16)	Allowed	Allowed	Allowed	Allowed	Allowed
READ DEFECT DATA (10)/(12)	Conflict	Conflict	Allowed	Conflict	Conflict
READ LONG (10)/(16)	Conflict	Conflict	Allowed	Conflict	Conflict
REASSIGN BLOCKS	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
SYNCHRONIZE CACHE (10)/(16)	Conflict	Conflict	Allowed	Conflict	Conflict
VERIFY (10)/(12)/(16)/(32)	Allowed	Conflict	Allowed	Allowed	Conflict
WRITE (6)/(10)/(12)/(16)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE AND VERIFY (10)/(12)/(16)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE LONG (10)/(16)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE SAME (10)/(16)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
XDREAD (10)/(32)	Allowed	Conflict	Allowed	Allowed	Conflict
XDWRITE (10)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
XDWRITEREAD (10)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
XPWRITE (10)/(32)	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 terminate the command with RESERVATION CONFLICT status.

4.13 Error reporting

4.13.1 Error reporting overview

If any of the conditions listed in table 3 occur during the processing of a command, the command shall be terminated 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-3 defines a deferred error reporting mechanism. Table 3 lists some error conditions and the applicable sense keys. The list does not provide an exhaustive enumeration of all conditions that may cause CHECK CONDITION status.

Table 3 — Example error conditions

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

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

Table 4 summarizes use of the sense data fields.

Table 4 — Sense data field usage for direct-access block devices

Field	Usage	Reference
VALID bit and INFORMATION field	READ LONG commands	5.14 and 5.15
	REASSIGN BLOCKS command	5.16
	WRITE LONG commands	5.33 and 5.34
	Any command that accesses the medium, based on the Read-Write Error Recovery mode page	6.3.4
COMMAND-SPECIFIC INFORMATION field	EXTENDED COPY command	SPC-3
	REASSIGN BLOCKS command	5.16
ILI bit	READ LONG commands	5.14 and 5.15
	WRITE LONG commands	5.33 and 5.34

When a command attempts to access or reference an invalid LBA, the first invalid LBA shall be returned in the INFORMATION field of the sense data (see SPC-3).

When a recovered read error is reported, the INFORMATION field of the sense data shall contain the LBA of the last recovered error during the transfer.

When an unrecovered read error is reported, the INFORMATION field of the sense data shall contain the LBA of the unrecovered logical block.

4.13.2 Block commands sense data descriptor

Table 5 defines the block commands sense data descriptor used in descriptor format sense data for direct-access block devices.

Table 5 — 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			ILI	Reserved			

The DESCRIPTOR TYPE field (see SPC-3) shall be set to 05h.

The ADDITIONAL LENGTH field (see SPC-3) shall be set to 02h.

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

4.14 Model for XOR commands

4.14.1 Model for XOR commands overview

In storage arrays, a storage array controller (see 3.1.41) organizes a group of direct-access block devices into objects. The type of object supported by this model is the redundancy group (see 3.1.37), where some of the logical blocks on the direct-access block devices are used for XOR-protected data (see 3.1.46) and some of the logical blocks are used for check data (see 3.1.5). The check data is generated by performing a cumulative XOR (see 3.1.18) operation of the XOR-protected data. The XOR operation may be performed by the storage array controller or by the direct-access block devices.

A direct-access block device containing XOR-protected data is called a data disk. A direct-access block device containing check data is called a parity disk.

Performing the XOR operation in the direct-access block devices may result in a reduced number of data transfers across the service delivery subsystem. For example, when the XOR operation is done within the storage array controller, four commands are needed for a typical update write sequence:

a) a command performing a read operation from the data disk;
b) a command performing a write operation to the data disk;
c) a command performing a read operation from the parity disk and
d) a command performing a write operation to the parity disk.

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

In contrast, during storage array controller supervised XOR operations (see 4.14.2) only three commands are needed:

a) a command performing a write operation to the data disk;
b) a command performing a read operation from the data disk and
c) a command performing a write operation to the parity disk.

4.14.2 Storage array controller supervised XOR operations

4.14.2.1 Storage array controller supervised XOR operations overview

A storage array controller supervises three basic operations that require XOR functionality:

a) update write operation (see 4.14.2.2);
b) regenerate operation (see 4.14.2.3); and
c) rebuild operation (see 4.14.2.4).

Command sequences for each of these operations use the device servers in the direct-access block devices to perform the necessary XOR functions.

Three XOR commands are needed to implement storage array controller supervised XOR operations: XDREAD commands (see 5.38 and 5.39), XDWRITE commands (see 5.40 and 5.41), and XPWRITE commands (see 5.44 and 5.45). An XDWRITEREAD command (see 5.42 and 5.43) may be used in place of a sequence of an XDWRITE command followed by an XDREAD command. The storage array controller also uses READ commands and WRITE commands for certain operations.

4.14.2.2 Update write operation

The update write operation writes new XOR-protected data to a data disk and updates the check data on the parity disk. The sequence is:

1) An XDWRITE command is sent to the data disk. This transfers new XOR-protected data to the data disk. The device server reads the old XOR-protected data, performs an XOR operation using the old XOR-protected data and the received XOR-protected data, retains the intermediate XOR result, and writes the received XOR-protected data to the medium;
2) An XDREAD command is sent to the data disk. This command transfers the intermediate XOR data to the storage array controller; and
3) An XPWRITE command is sent to the parity disk. This transfers the intermediate XOR data (i.e., XOR data received in a previous XDREAD command) to the parity disk. The device server reads the old check data, performs an XOR operation using the old check data and the intermediate XOR data, and writes the result (i.e., the new check data) to the medium.

In place of steps 1) and 2), a single XDWRITEREAD command may be sent.

4.14.2.3 Regenerate operation

The regenerate operation is used to recreate one or more logical blocks that have an error. This is accomplished by reading the associated logical block from each of the other direct-access block devices within the redundancy group and performing an XOR operation with each of these logical blocks. The last XOR result is the data that should have been present on the unreadable direct-access block device. The number of steps is dependent on the number of direct-access block devices in the redundancy group, but the sequence is as follows:

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

The intermediate XOR data returned by the last XDREAD command is the regenerated data for the failed device.

In place of steps 2) and 3), a single XDWRITEREAD command with the DISABLE WRITE bit set to one may be used.

4.14.2.4 Rebuild operation

The rebuild operation is similar to the regenerate operation, except that the last XOR result is written to the replacement device. This function is used when a failed device is replaced and the storage array controller is writing the rebuilt data to the replacement device. The number of steps is dependent on the number of direct-access block devices in the redundancy group, but the sequence is as follows:

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

In place of steps 2) and 3), a single XDWRITEREAD command with the DISABLE WRITE bit set to one may be used.

4.14.3 Array subsystem considerations

4.14.3.1 Array subsystem considerations overview

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

4.14.3.2 Buffer full status handling

When the storage array controller sends an XDWRITE command to a device, the device retains the resulting XOR data until the storage array controller issues a matching XDREAD command to retrieve the data (see 4.14.4). Depending on the size of the device's buffer and the size of the XOR data, this may consume all of the device's internal buffer space. When all of the device's internal buffer space is allocated for XOR data, it may not be able to accept new medium access commands other than valid XDREAD commands and it may not be able to begin processing of commands that are already in the task set.

When the device server is not able to accept a new command because there is not enough space in the buffer, 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 BUFFER FULL.

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

The bidirectional XDWRITEREAD command avoids the buffer full condition, since the device server controls when it accepts more write data and provides read data.

4.14.3.3 Access to an inconsistent stripe

A stripe is a set of corresponding extents (see 3.1.19) from two or more direct-access block devices.

When the storage array controller issues an update write to a data disk, the data in the data disk has been updated when successful status is returned for the command. Until the parity disk has been updated, however, the associated stripe in the redundancy group is not consistent (i.e., performing an XOR operation on the XOR-protected data does not produce the check data).

The storage array controller shall keep track of this window of inconsistency and ensure that a regenerate or rebuild operation for any data extent within the stripe is not attempted until after the parity disk has been updated, making the stripe consistent again. For multi-initiator systems, tracking the updates may be more complex because each storage array controller needs to ensure that a second storage array controller is not writing to a stripe that the first storage array controller is regenerating or rebuilding. The coordination between storage array controllers is system specific and is beyond the scope of this standard.

If any of the XOR commands end with CHECK CONDITION status and an unrecovered error is indicated, an inconsistent stripe may result. It is the storage array controller's responsibility to identify the failing device, to identify the scope of the failure and then limit access to the inconsistent stripe. The recovery procedures that the storage array controller implements are outside the scope of this standard.

4.14.4 XOR data retention requirements

The device server shall retain XOR data resulting from an XDWRITE command awaiting retrieval by a matching XDREAD command until one of the following events occurs:

a) a matching XDREAD command;
b) logical unit reset;
c) I_T nexus loss associated with the I_T nexus that sent the XDWRITE command;
d) processing any of the following task management functions (see SAM-3):
   A) CLEAR TASK SET;
   B) ABORT TASK specifying the I_T_L_Q nexus of an XDREAD command retrieving that XOR data; or
   C) ABORT TASK SET.

If the XOR data is lost and the application client still wants to perform the XOR operation, it is required to resend the XDWRITE command after one of those events.

4.15 START STOP UNIT and power conditions

4.15.1 START STOP UNIT and power conditions overview

The START STOP UNIT command (see 5.17) allows an application client to control the power condition of a logical unit. This method includes specifying that the logical unit transitions to a power condition.

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

There shall be no notification to the application client that a logical unit has transitioned from one power condition to another. An application client may determine the current power condition of a logical unit by issuing a REQUEST SENSE command (see SPC-3). The device server returns parameter data with the sense key set to NO SENSE and the additional sense code set to one of the following:

a) LOW POWER CONDITION ON if the reason for entry into the standby power condition or idle power condition is unknown;
b) IDLE CONDITION ACTIVATED BY TIMER if the logical unit entered the idle power condition due to the idle condition timer (see SPC-3);
c) STANDBY CONDITION ACTIVATED BY TIMER if the logical unit entered the standby power condition due to the idle condition timer (see SPC-3);
d) IDLE CONDITION ACTIVATED BY COMMAND if the logical unit entered the idle power condition due to a START STOP UNIT command or
e) STANDBY CONDITION ACTIVATED BY COMMAND if the logical unit entered the standby power condition due to a START STOP UNIT command.

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

4.15.2 START STOP UNIT and power conditions state machine

4.15.2.1 START STOP UNIT and power conditions state machine overview

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

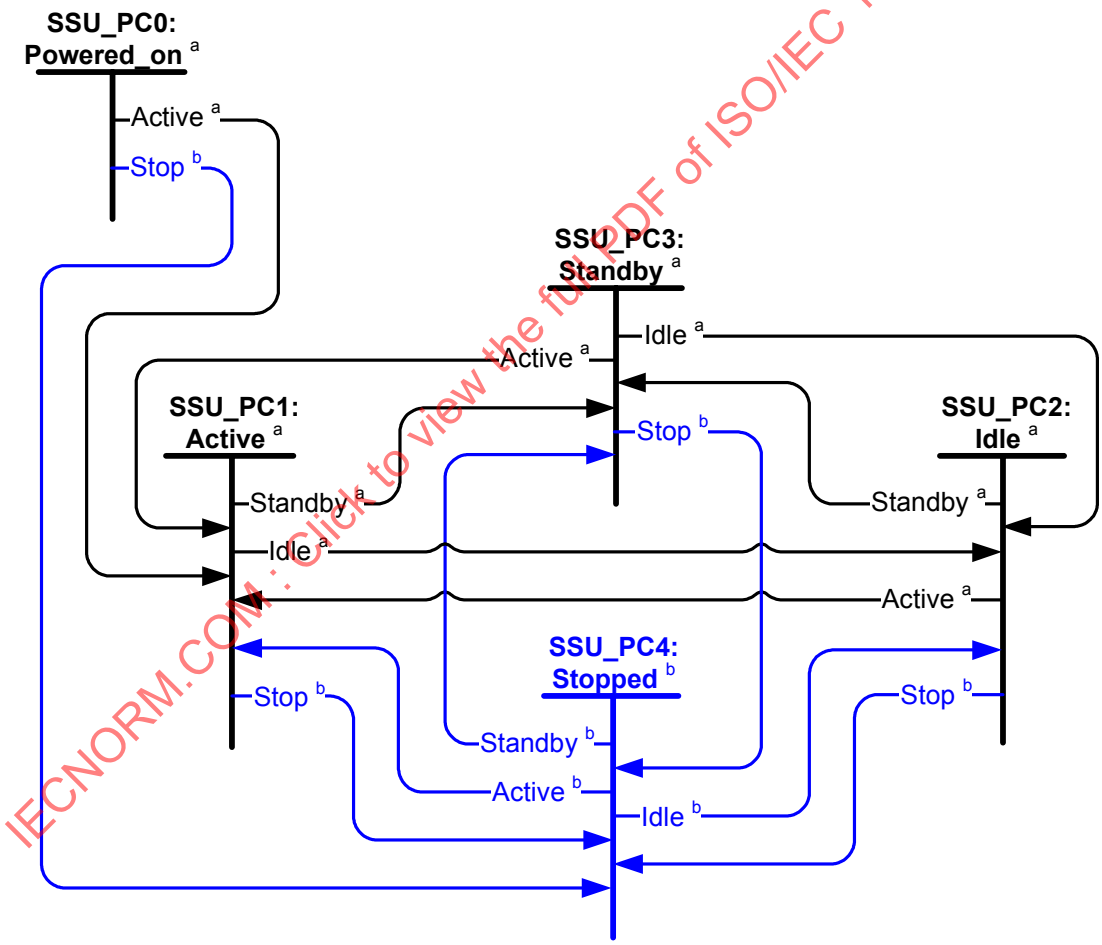
The SSU_PC states are as follows:

a) SSU_PC0:Powered_on (see 4.15.2.2) (initial state);
b) SSU_PC1:Active (see 4.15.2.3);
c) SSU_PC2:Idle (see 4.15.2.4);
d) SSU_PC3:Standby (see 4.15.2.5); and
e) SSU_PC4:Stopped (see 4.15.2.6).

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

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

Figure 1 describes the SSU_PC state machine.



Notes:
[a] This state or transition is also described in SPC-3, but may have additional characteristics unique to this standard (e.g., a transition to or from a state described in this standard).
[b] This state or transition is described in this standard.

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

4.15.2.2 SSU_PC0:Powered_on state

4.15.2.2.1 SSU_PC0:Powered_on state description

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

4.15.2.2.2 Transition SSU_PC0:Powered_on to SSU_PC1:Active

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

4.15.2.2.3 Transition SSU_PC0:Powered_on to SSU_PC4:Stopped

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

4.15.2.3 SSU_PC1:Active state

4.15.2.3.1 SSU_PC1:Active state description

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

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

a) the logical unit is in the active power condition (see SPC-3);
b) if the idle condition timer is active (see SPC-3) and not disabled (see 5.17), then the idle condition timer is running; and
c) if the standby condition timer is active (see SPC-3) and not disabled (see 5.17), then the standby condition timer is running.

4.15.2.3.2 Transition SSU_PC1:Active to SSU_PC2:Idle

This transition shall occur after:

a) the device server receives a START STOP UNIT command with the POWER CONDITION field set to IDLE;
b) the device server receives a START STOP UNIT command with the POWER CONDITION field set to FORCE_IDLE_0; or
c) the idle condition timer is active (see SPC-3), enabled (see 5.17) and zero.

4.15.2.3.3 Transition SSU_PC1:Active to SSU_PC3:Standby

This transition shall occur after:

a) the device server receives a START STOP UNIT command with the POWER CONDITION field set to STANDBY;
b) the device server receives a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0; or
c) the standby condition timer is active (see SPC-3), enabled (see 5.17), and zero.

4.15.2.3.4 Transition SSU_PC1:Active to SSU_PC4:Stopped

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

4.15.2.4 SSU_PC2:Idle state

4.15.2.4.1 SSU_PC2:Idle state description

While in this state:

a) the logical unit is in the idle power condition (see SPC-3) and
b) if the standby condition timer is active (see SPC-3) and not disabled (see 5.17), then the standby condition timer is running.

4.15.2.4.2 Transition SSU_PC2:Idle to SSU_PC1:Active

This transition shall occur after:

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

4.15.2.4.3 Transition SSU_PC2:Idle to SSU_PC3:Standby

This transition shall occur after:

a) the device server receives a START STOP UNIT command with the POWER CONDITION field set to STANDBY;
b) the device server receives a START STOP UNIT command with the POWER CONDITION field set to FORCE_STANDBY_0; or
c) the standby condition timer is active (see SPC-3), enabled (see 5.17), and zero.

4.15.2.4.4 Transition SSU_PC2:Idle to SSU_PC4:Stopped

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

4.15.2.5 SSU_PC3:Standby state

4.15.2.5.1 SSU_PC3:Standby state description

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

4.15.2.5.2 Transition SSU_PC3:Standby to SSU_PC1:Active

This transition shall occur after:

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

4.15.2.5.3 Transition SSU_PC3:Standby to SSU_PC2:Idle

This transition shall occur after:

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

4.15.2.5.4 Transition SSU_PC3:Standby to SSU_PC4:Stopped

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

4.15.2.6 SSU_PC4:Stopped state

4.15.2.6.1 SSU_PC4:Stopped state description

While in this state:

a) the device server is not capable of processing medium access commands. Any medium access commands received while in this state shall cause the device server to terminate the command with 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 and
b) the power consumed by the SCSI target device should be less than or equal to that consumed when the logical unit is in the SSU_PC1:Active, SSU_PC2:Idle, or SSU_PC3:Standby states.

4.15.2.6.2 Transition SSU_PC4:Stopped to SSU_PC1:Active

This transition shall occur after:

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

4.15.2.6.3 Transition SSU_PC4:Stopped to SSU_PC2:Idle

This transition shall occur after:

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

4.15.2.6.4 Transition SSU_PC4:Stopped to SSU_PC3:Standby

This transition shall occur after:

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

4.16 Protection information model

4.16.1 Protection information overview

The protection information model provides for protection of user data while it is being transferred between a sender and a receiver. Protection information is generated at the application layer and may be checked by any object associated with the I_T_L nexus. Once received, protection information is retained (e.g., written to 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-3).

For commands that are using protection information, the data-in buffer and/or data-out buffer shall consist of logical blocks with both user data and protection information. For commands that are not using protection information, the data-in buffer and/or data-out buffer shall consist of logical blocks with only user data.

If the logical unit is formatted with protection information and the EMDP bit is set to one in the Disconnect-Reconnect mode page (see SPC-3), then checking of the logical block reference tag within the service delivery subsystem without accounting for modified data pointers and data alignments may cause false errors when logical blocks are transmitted out of order.

4.16.2 Protection information format

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

Table 6 — User data and protection information format

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

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

The LOGICAL BLOCK GUARD field contains the CRC (see 4.16.3) of the contents of the USER DATA field.

The LOGICAL BLOCK APPLICATION TAG field is set by the application client. A LOGICAL BLOCK APPLICATION TAG field set to FFFFh disables checking of all protection information for the logical block when reading from the medium. Otherwise, the contents of the logical block application tag are not defined by this standard. The LOGICAL BLOCK APPLICATION TAG field may be modified by a device server if the ATO bit is set to zero in the Control mode page (see SPC-3). The contents of the LOGICAL BLOCK APPLICATION TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

The LOGICAL BLOCK REFERENCE TAG field is an incrementing value associated with the logical block. The LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer depends on the command being processed:

a) for a command that does not include an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field (e.g., READ (16)), the LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer shall contain the least significant four bytes of the LBA contained in the LOGICAL BLOCK ADDRESS field of the command; and
b) for a command that does include an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field (e.g., READ (32)), the LOGICAL BLOCK REFERENCE TAG field of the first logical block shall contain the value in the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field of the command. These commands are only processed if the medium was formatted with application client ownership of the logical block reference tag (i.e., with the RTO_REQ bit set to one in the FORMAT UNIT command (see 5.2)).

Each subsequent logical block in the data-in buffer and/or data-out buffer shall contain a LOGICAL BLOCK REFERENCE TAG field with the logical block reference tag of the previous logical block plus one. The contents of the LOGICAL BLOCK REFERENCE TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

4.16.3 Logical block guard

4.16.3.1 Logical block guard overview

The LOGICAL BLOCK GUARD field shall contain a CRC that is generated from the contents of the USER DATA field.

Table 7 defines the CRC polynomials used to generate the logical block guard from the contents of the USER DATA field.

Table 7 — 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., G(x) = 18BB7h)
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.

4.16.3.2 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)):

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

4.16.3.3 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.

4.16.3.4 CRC test cases

Several CRC test cases are shown in table 8.

Table 8 — CRC test cases

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

4.16.4 Application of protection information

Before an application client transmits or receives logical blocks with protection information it shall:

1) determine if a logical unit supports protection information using the INQUIRY command (see the PROTECT bit in the standard INQUIRY data in SPC-3);
2) if protection information is supported, then determine if the logical unit is formatted to accept protection information using the READ CAPACITY (16) command (see the PROT_EN bit in 5.11); and
3) if the logical unit supports protection information and is not formatted to accept protection information, then format the logical unit with protection information enabled.

If the logical unit supports protection information and is formatted to accept protection information, then the application client may use commands performing read operations that support protection information and should use commands performing write and verify operations that support protection information.

4.16.5 Protection information and commands

The enabling of protection information enables fields in some 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 when protection information is enabled are listed in table 9 (see 5.1).

Commands that return the length in bytes of each logical block (e.g., the MODE SENSE commands and the READ CAPACITY commands) shall return the length of the USER DATA field and shall not include the length of the protection information (i.e., the LOGICAL BLOCK GUARD field, the LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK REFERENCE TAG field) (e.g., if the user data plus the protection information is equal to 520 bytes then 512 is returned).

4.17 Grouping function

A grouping function is a function that collects information about attributes associated with commands (i.e., information about commands with the same group value are collected into the specified group). The definition of the attributes and the groups is outside the scope of this standard. Groups are identified with the GROUP NUMBER field in the CDB of certain commands (e.g., the PRE-FETCH (10) command (see 5.3)).

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

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

Support for the grouping function is indicated in the GROUP_SUP bit in the Extended INQUIRY Data VPD page (see SPC-3).

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 9. Commands with CDB or parameter data fields that support protection information (see 4.16) or for which protection information may be a factor in the processing of the command are indicated by the fourth (i.e, Protection information) column.

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

Command name	Operation code [a]	Type [b]	Protection information	Reference
ACCESS CONTROL IN	86h	O	no	SPC-3
ACCESS CONTROL OUT	87h	O	no	SPC-3
CHANGE ALIASES	A4h/0Bh	O	no	SPC-3
EXTENDED COPY	83h	O	no	SPC-3
FORMAT UNIT	04h	M	yes	5.2
INQUIRY	12h	M	yes	SPC-3
LOG SELECT	4Ch	O	no	SPC-3
LOG SENSE	4Dh	O	no	SPC-3
MAINTENANCE IN	A3h/00h - 04h A3h/06h - 09h	X [e]	no	SCC-2
MAINTENANCE OUT	A4h/00h - 05h A4h/07h - 09h	X [e]	no	SCC-2
MODE SELECT (6)	15h	O	no	SPC-3
MODE SELECT (10)	55h	O	no	SPC-3
MODE SENSE (6)	1Ah	O	no	SPC-3
MODE SENSE (10)	5Ah	O	no	SPC-3
MOVE MEDIUM ATTACHED	A7h	X [f]	no	SMC-2
PERSISTENT RESERVE IN	5Eh	O	no	SPC-3
PERSISTENT RESERVE OUT	5Fh	O	no	SPC-3
PRE-FETCH (10)	34h	O	no	5.3
PRE-FETCH (16)	90h	O	no	5.4
PREVENT ALLOW MEDIUM REMOVAL	1Eh	O	no	SPC-3
READ (6)	08h	M [c]	yes	5.5
READ (10)	28h	M	yes	5.6
READ (12)	A8h	O	yes	5.7
READ (16)	88h	O	yes	5.8
READ (32)	7Fh/0009h	O	yes	5.9
READ ATTRIBUTE	8Ch	O	no	SPC-3
READ BUFFER	3Ch	O	no	SPC-3
READ CAPACITY (10)	25h	M	no	5.10
READ CAPACITY (16)	9Eh/10h	X [d]	yes	5.11

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

Command name	Operation code [a]	Type [b]	Protection information	Reference
READ DEFECT DATA (10)	37h	O	no	5.12
READ DEFECT DATA (12)	B7h	O	no	5.13
READ ELEMENT STATUS ATTACHED	B4h	X [f]	no	SMC-2
READ LONG (10)	3Eh	O	yes	5.14
READ LONG (16)	9Eh/11h	O	yes	5.15
REASSIGN BLOCKS	07h	O	no	5.16
RECEIVE COPY RESULTS	84h	O	no	SPC-3
RECEIVE DIAGNOSTIC RESULTS	1Ch	O/M [g]	no	SPC-3
REDUNDANCY GROUP IN	BAh	X [e]	no	SCC-2
REDUNDANCY GROUP OUT	BBh	X [e]	no	SCC-2
REPORT ALIASES	A3h/0Bh	O	no	SPC-3
REPORT DEVICE IDENTIFIER	A3h/05h	O	no	SPC-3
REPORT LUNS	A0h	M	no	SPC-3
REPORT PRIORITY	A3h/0Eh	O	no	SPC-3
REPORT SUPPORTED OPERATION CODES	A3h/0Ch	O	no	SPC-3
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS	A3h/0Dh	O	no	SPC-3
REPORT TARGET PORT GROUPS	A3h/0Ah	O	no	SPC-3
REQUEST SENSE	03h	M	no	SPC-3
SEND DIAGNOSTIC	1Dh	M	no	SPC-3
SET DEVICE IDENTIFIER	A4h/06h	O	no	SPC-3
SET PRIORITY	A4h/0Eh	O	no	SPC-3
SET TARGET PORT GROUPS	A4h/0Ah	O	no	SPC-3
SPARE IN	BCh	X [e]	no	SCC-2
SPARE OUT	BDh	X [e]	no	SCC-2
START STOP UNIT	1Bh	O	no	5.17
SYNCHRONIZE CACHE (10)	35h	O	no	5.18
SYNCHRONIZE CACHE (16)	91h	O	no	5.19
TEST UNIT READY	00h	M	no	SPC-3
VERIFY (10)	2Fh	O	yes	5.20
VERIFY (12)	AFh	O	yes	5.21
VERIFY (16)	8Fh	O	yes	5.22
VERIFY (32)	7Fh/000Ah	O	yes	5.23
VOLUME SET IN	BEh	X [e]	no	SCC-2
VOLUME SET OUT	BFh	X [e]	no	SCC-2
WRITE (6)	0Ah	O [c]	yes	5.24

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

Command name	Operation code [a]	Type [b]	Protection information	Reference
WRITE (10)	2Ah	O	yes	5.25
WRITE (12)	AAh	O	yes	5.26
WRITE (16)	8Ah	O	yes	5.27
WRITE (32)	7Fh/000Bh	O	yes	5.28
WRITE AND VERIFY (10)	2Eh	O	yes	5.29
WRITE AND VERIFY (12)	AEh	O	yes	5.30
WRITE AND VERIFY (16)	8Eh	O	yes	5.31
WRITE AND VERIFY (32)	7Fh/000Ch	O	yes	5.32
WRITE ATTRIBUTE	8Dh	O	no	SPC-3
WRITE BUFFER	3Bh	O	no	SPC-3
WRITE LONG (10)	3Fh	O	yes	5.33
WRITE LONG (16)	9Fh/11h	O	yes	5.34
WRITE SAME (10)	41h	O	yes	5.35
WRITE SAME (16)	93h	O	yes	5.36
WRITE SAME (32)	7Fh/000Dh	O	yes	5.37
XDREAD (10)	52h	O	yes	5.38
XDREAD (32)	7Fh/0003h	O	yes	5.39

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

Command name	Operation code [a]	Type [b]	Protection information	Reference
XDWRITE (10)	50h	O	yes	5.40
XDWRITE (32)	7Fh/0004h	O	yes	5.41
XDWRITEREAD (10)	53h	O	yes	5.42
XDWRITEREAD (32)	7Fh/0007h	O	yes	5.43
XPWRITE (10)	51h	O	yes	5.44
XPWRITE (32)	7Fh/0006h	O	yes	5.45
The following operation codes are obsolete: 01h (REZERO UNIT), 0Bh (SEEK (6)), 16h (RESERVE (6)), 17h (RELEASE (6)), 18h (COPY), 2Bh (SEEK (10)), 30h (SEARCH DATA HIGH (10)), 31h (SEARCH DATA EQUAL (10)), 32h (SEARCH DATA LOW (10)), 33h (SET LIMITS (10)), 36h (LOCK UNLOCK CACHE (10)), 39h (COMPARE), 3Ah (COPY AND VERIFY), 40h (CHANGE DEFINITION), 56h (RESERVE (10)), 57h (RELEASE (10)), 80h (XDWRITE EXTENDED (16)), 81h (REBUILD (16)), 82h (REGENERATE (16)), 92h (LOCK UNLOCK CACHE (16)), and B3h (SET LIMITS (12)). The following operation codes are vendor-specific: 02h, 05h, 06h, 09h, 0Ch, 0Dh, 0Eh, 0Fh, 10h, 11h, 13h, 14h, 19h, 20h, 21h, 22h, 23h, 24h, 26h, 27h, 29h, 2Ch, 2Dh and C0h through FFh. All operation codes for direct-access block devices not specified in this table are reserved for future standardization.				

[a] Some commands are defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.
[b] M = command implementation is mandatory. O = command implementation is optional. X = Command implementation requirements are detailed in the reference.
[c] Application clients should migrate from READ (6) to READ (10) (see 5.5) and from WRITE (6) to WRITE (10) (see 5.24).
[d] READ CAPACITY (16) is mandatory if protection information is supported and optional otherwise.
[e] If the SCCS bit is set to one in the standard INQUIRY data (see SPC-3), these commands shall be supported as required by SCC-2. If the SCCS bit is set to zero, these commands shall not be supported.
[f] If the MCHGR bit is set to one in the standard INQUIRY data (see SPC-3), these commands shall be supported as required in SMC-2. If the MCHGR bit is set to zero, these commands shall not be supported.
[g] This command shall be supported if the ENCSERV bit is set to one in the standard INQUIRY data (see SPC-3) and may be supported otherwise.

5.2 FORMAT UNIT command

5.2.1 FORMAT UNIT command overview

The FORMAT UNIT command (see table 10) requests that the device server format the medium into application client accessible logical blocks as specified in the number of blocks and block length values received in the last mode parameter block descriptor (see 6.3.2) in a MODE SELECT command (see SPC-3). In addition, the device server may certify the medium and create control structures for the management of the medium and defects. The degree that the medium is altered by this command is vendor-specific.

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

Table 10 — FORMAT UNIT command

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

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

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

While performing a format operation, the device server shall respond to commands attempting to enter into the task set 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. Handling of commands already in the task set is vendor-specific.

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

A format protection information (FMTPINFO) bit set to zero specifies that the device server shall disable the use of protection information (see 4.16) and format the medium to the block length specified in the mode parameter block descriptor of the mode parameter header (see SPC-3). A FMTPINFO bit set to one specifies that the device server shall enable the use of protection information (see 4.16) and format the medium to the block length specified in the mode parameter block descriptor of the mode parameter header plus eight (e.g., if the block length is 512, then the formatted block length is 520). Following a successful format, the PROT_EN bit in the READ CAPACITY (16) parameter data (see 5.11) indicates whether protection information (see 4.16) is enabled.

The reference tag own request (RTO_REQ) bit specifies whether the application client or the device server has ownership of the LOGICAL BLOCK REFERENCE TAG field in protection information (see 4.16.2). If the FMTPINFO bit is set to zero and the RTO_REQ bit is set to one, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. If the FMTPINFO bit is set to one and the RTO_REQ bit is set to one, the device server shall enable application client ownership of the LOGICAL BLOCK REFERENCE TAG field. If the FMTPINFO bit set to one and the RTO_REQ bit is set to zero, the device server shall disable application client ownership (i.e., enable device server ownership) of the LOGICAL BLOCK REFERENCE TAG field. Following a successful format,

the RTO_EN bit in the READ CAPACITY (16) parameter data (see 5.11) indicates whether application client ownership of the LOGICAL BLOCK REFERENCE TAG field is enabled.

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

A LONGLIST bit set to zero specifies that the parameter list, if any, contains a short parameter list header as defined in table 13. A LONGLIST bit set to one specifies that the parameter list, if any, contains a long parameter list header as defined in table 14. If the FMTDATA bit is set to zero, the LONGLIST bit shall be ignored.

A format data (FMTDATA) bit set to zero specifies that no parameter list be transferred from the data-out buffer.

A FMTDATA bit set to one specifies that the FORMAT UNIT parameter list (see table 12) shall be transferred from the data-out buffer. The parameter list consists of a parameter list header, followed by an optional initialization pattern descriptor, followed by an optional defect list.

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

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

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

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

If the FMTDATA bit is set to zero, the CMPLIST bit shall be ignored.

The DEFECT LIST FORMAT field specifies the format of the address descriptors in the defect list if the FMTDATA bit is set to one (see table 11).

Table 11 defines the address descriptor usage for the FORMAT UNIT command.

Table 11 — FORMAT UNIT command address descriptor usage

Field in the FORMAT UNIT CDB			DEFECT LIST LENGTH **field in the parameter list header**	**Type** [a]	**Comments** [f]
FMTDATA	CMPLST	DEFECT LIST FORMAT			
0	any	000b	Not available	M	Vendor-specific defect information
1	0	000b (short block)	Zero	O	See [b] and [d]
1	1			O	See [b] and [e]
1	0		Nonzero	O	See [c] and [d]
1	1			O	See [b] and [e]
		011b (long block)	Zero	O	See [b] and [d]
				O	See [b] and [e]
1	0		Nonzero	O	See [c] and [d]
1	1			O	See [c] and [e]
1	0	100b (bytes from index)	Zero	O	See [b] and [d]
1	1			O	See [b] and [e]
1	0		Nonzero	O	See [c] and [d]
1	1			O	See [c] and [e]
1	0	101b (physical sector)	Zero	O	See [b] and [d]
1	1			O	See [b] and [e]
1	0		Nonzero	O	See [c] and [d]
1	1			O	See [c] and [e]
1	0	110b (vendor specific)	Vendor specific	O	
1	1			O	
All others				Reserved.	

[a] M = implementation is mandatory. O = implementation is optional.
[b] No DLIST is included in the parameter list.
[c] A DLIST is included in the parameter list. The device server shall add the DLIST defects to the new GLIST.
[d] The device server shall add existing GLIST defects to the new GLIST (i.e., use the existing GLIST).
[e] The device server shall not add existing GLIST defects to the new GLIST (i.e., discard the existing GLIST).
[f] All the options described in this table cause a new GLIST to be created during processing of the FORMAT UNIT command as described in the text.

5.2.2 FORMAT UNIT parameter list

5.2.2.1 FORMAT UNIT parameter list overview

Table 12 defines the FORMAT UNIT parameter list.

Table 12 — FORMAT UNIT parameter list

Byte\Bit	7	6	5	4	3	2	1	0
0 to 3 or 0 to 7	Parameter list header (see table 13 or table 14 in 5.2.2.2)							
	Initialization pattern descriptor (if any)(see table 15 in 5.2.2.3)							
	Defect list (if any)							

The parameter list header is defined in 5.2.2.2.

The initialization pattern descriptor, if any, is defined in 5.2.2.3.

The defect list, if any, contains address descriptors (see 5.2.2.4) each specifying a location on the medium that the device server shall exclude from the application client accessible part. This is called the DLIST (see 4.8).

5.2.2.2 Parameter list header

The parameter list headers (see table 13 and table 14) provide several optional format control parameters. Device servers that implement these headers provide the application client additional control over the use of the four defect sources, and the format operation. If the application client attempts to select any function not implemented by the device server, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The short parameter list header (see table 13) is used if the LONGLIST bit is set to zero in the FORMAT UNIT CDB.

Table 13 — Short parameter list header

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

The long parameter list header (see table 14) is used if the LONGLIST bit is set to one in the FORMAT UNIT CDB.

Table 14 — Long parameter list header

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

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

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

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

A DPRY bit set to one specifies that the device server shall not use the PLIST to identify defective areas of the medium. The PLIST shall not be deleted.

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

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

a) the device server has been requested to use the PLIST (i.e., the DPRY bit is set to zero) or the GLIST (i.e., the CMPLST bit is set to zero) and the device server is not able to locate the list or determine whether the list exists or
b) the device server has been requested to use the PLIST (i.e., the DPRY bit is set to zero) or the GLIST (i.e., the CMPLST bit is set to zero) and the device server encounters an error while accessing the defect list.

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

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

NOTE The use of the FMTDATA bit, the CMPLST bit, and the parameter list header allow the application client to control the source of the defect lists used by the FORMAT UNIT command. Setting the DEFECT LIST LENGTH field to zero allows the application client to control the use of PLIST and CLIST without having to specify a DLIST.

An initialization pattern (IP) bit set to zero specifies that an initialization pattern descriptor is not included and that the device server shall use its default initialization pattern. An IP bit set to one specifies that an initialization pattern descriptor (see 5.2.2.3) is included in the FORMAT UNIT parameter list following the parameter list header.

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

The DEFECT LIST LENGTH field specifies the total length in bytes of the defect list (i.e., the address descriptors) that follows and does not include the initialization pattern descriptor, if any. The formats for the address descriptor(s) are shown in 5.2.2.4.

Short block format address descriptors and long block format address descriptors should be in ascending order. Bytes from index format address descriptors and physical sector format address descriptors shall be in ascending order. More than one physical or logical block may be affected by each address descriptor. If the address descriptors are not in the required order, the device server shall 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.

5.2.2.3 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 15) is sent to the device server as part of the FORMAT UNIT parameter list.

Table 15 — Initialization pattern descriptor

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

The initialization pattern modifier (IP MODIFIER) field (see table 16) specifies the type and location of a header that modifies the initialization pattern.

Table 16 — Initialization pattern modifier (IP MODIFIER) field

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

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

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

When the SI bit is set to one, the device server need not write the initialization pattern over the header and other 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), it 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.

The INITIALIZATION PATTERN TYPE field (see table 17) 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. The initialization pattern is modified by the IP MODIFIER field as described in table 16.

Table 17 — 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 - 7Fh	Reserved
80h - FFh	Vendor-specific

[a] If the INITIALIZATION PATTERN LENGTH field is not set to zero, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

[b] If the INITIALIZATION PATTERN LENGTH field is set to zero, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER 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 block length the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The INITIALIZATION PATTERN field specifies the initialization pattern. The initialization pattern is modified by the IP MODIFIER field.

5.2.2.4 Address descriptor formats

5.2.2.4.1 Address descriptor formats overview

This subclause describes the address descriptor formats used in the FORMAT UNIT command, the READ DEFECT DATA commands (see 5.12 and 5.13), and the Translate Address diagnostic pages (see 6.1.2 and 6.1.3) of the SEND DIAGNOSTIC command and the RECEIVE DIAGNOSTIC RESULTS command.

The format type of an address descriptor is specified with:

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

Table 18 defines the types of address descriptors.

Table 18 — Address descriptor formats

Format type	Description	Reference
000b	Short block format address descriptor	5.2.2.4.2
011b	Long block format address descriptor	5.2.2.4.3
100b	Bytes from index format address descriptor	5.2.2.4.4
101b	Physical sector format address descriptor	5.2.2.4.5
110b	Vendor-specific	
All others	Reserved	

5.2.2.4.2 Short block format address descriptor

A format type of 000b specifies the short block format address descriptor defined in table 19.

Table 19 — Short block format address descriptor (000b)

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

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

5.2.2.4.3 Long block format address descriptor

A format type of 011b specifies the long block format address descriptor defined in table 20.

Table 20 — Long block format address descriptor (011b)

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

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

5.2.2.4.4 Bytes from index format address descriptor

A format type of 100b specifies the bytes from index address descriptor defined in table 21. For the FORMAT UNIT command and the READ DEFECT DATA commands, this descriptor specifies the location of a defect

that is either the length of one track or is no more than eight bytes long. For the Translate Address diagnostic pages, this descriptor specifies the location of a track or the first byte or last byte of an area.

Table 21 — Bytes from index format address descriptor (100b)

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

The CYLINDER NUMBER field contains the cylinder number.

The HEAD NUMBER field contains the head number.

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

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

5.2.2.4.5 Physical sector format address descriptor

A format type of 101b specifies the physical sector address descriptor defined in table 22. For the FORMAT UNIT command and the READ DEFECT DATA commands, this descriptor specifies the location of a defect that is either the length of one track or the length of one sector. For the Translate Address diagnostic pages, this descriptor specifies the location of a track or a sector.

Table 22 — Physical sector format address descriptor (101b)

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

The CYLINDER NUMBER field contains the cylinder number.

The HEAD NUMBER field contains the head number.

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

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

5.3 PRE-FETCH (10) command

The PRE-FETCH (10) command (see table 23) requests that the device server transfer the specified logical blocks from the medium to the volatile cache and/or non-volatile cache. Logical blocks include user data and,

if the medium is formatted with protection information enabled, protection information. No data shall be transferred to the data-in buffer.

Table 23 — PRE-FETCH (10) command

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

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

The LOGICAL BLOCK ADDRESS field specifies the first logical block accessed by this command. If the logical block address exceeds the capacity of the medium 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 GROUP NUMBER field specifies the group into which attributes associated with the command should be collected (see 4.17). 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 logical block specified by the LOGICAL BLOCK ADDRESS field. A PREFETCH LENGTH field set to zero specifies that all logical blocks starting with the one 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 logical block address plus the prefetch length exceeds the capacity of the medium, 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 device server is not required to transfer logical blocks that already are contained in the cache.

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

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

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

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

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

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

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

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

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

5.4 PRE-FETCH (16) command

The PRE-FETCH (16) command (see table 24) requests that the device server transfer the specified logical blocks from the medium to the volatile cache and/or non-volatile cache. Logical blocks include user data and, if the medium is formatted with protection information enabled, protection information. No data shall be transferred to the data-in buffer.

Table 24 — PRE-FETCH (16) command

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

See the PRE-FETCH (10) command (see 5.3) for the definitions of the fields in this command.

5.5 READ (6) command

The READ (6) command (see table 25) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes user data but does not include protection information. The most recent data value written, or to be written if cached, in the addressed logical blocks shall be returned.

Table 25 — READ (6) command

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

The cache control bits (see 5.6) are not provided for this command. Direct-access block devices with cache may have values for the cache control bits that affect the READ (6) command; however, no default values are defined by this standard. If explicit control is required, the READ (10) command should be used.

See the PRE-FETCH (10) command (see 5.3) for the definition of the LOGICAL BLOCK ADDRESS 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 specified by the LOGICAL BLOCK ADDRESS field. A TRANSFER LENGTH field set to zero specifies that 256 logical blocks shall be read. Any other value specifies the number of logical blocks that shall be read. If the logical block address plus the transfer length exceeds the capacity of the medium, 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 in the Block Limits VPD page (see 6.4.2).

NOTE 1 For the READ (10) command, READ (12) command, READ (16) command, and READ (32) command, a TRANSFER LENGTH field set to zero specifies that no logical blocks are read.

NOTE 2 Although the READ (6) command is limited to addressing logical blocks up to a capacity of 1 GiB (i.e., 1 073 741 824 bytes)(see IEC 60027-2:2000), for block lengths of 512 bytes, this command has been maintained as mandatory since some system initialization routines require that the READ (6) command be used. System initialization routines should migrate from the READ (6) command to the READ (10) command, which is capable of addressing 2 TiB (i.e., 2 199 023 255 552 bytes) with block lengths of 512 bytes or the READ (16) command to address more than 2 TiB.

The device server shall check the protection information read from the medium before returning status for the command as described in table 26.

Table 26 — Protection information checking for READ (6)

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

[a] The device server checks the logical block application tag only if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. The method for acquiring this knowledge is not defined by this standard.
[b] If an error is reported, the sense key shall be set to ABORTED COMMAND.
[c] If multiple errors occur, the selection of which error to report is not defined by this standard.
[d] See the Extended INQUIRY Data VPD page (see SPC-3) for the definitions of the GRD_CHK bit, APP_CHK bit, and REF_CHK bit.
[e] If the device server detects a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, it shall not check any protection information in the associated logical block.
[f] If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11), the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If the RTO_EN bit is set to one, the device server checks the logical block reference tag only if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. The method for acquiring this knowledge is not defined in this standard.

5.6 READ (10) command

The READ (10) command (see table 27) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes

user data and may include protection information, based on the RDPROTECT field and the medium format. The most recent data value written in the addressed logical block shall be returned.

Table 27 — READ (10) command

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

See the PRE-FETCH (10) command (see 5.3) for the definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command (see 5.3) and 4.17 for the definition of the GROUP NUMBER field.

The device server shall check the protection information read from the medium before returning status for the command based on the RDPROTECT field as described in table 28.

Table 28 — RDPROTECT field (Sheet 1 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
000b [i]	Yes	No	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
				GRD_CHK = 0	No check performed
			LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
				APP_CHK = 0	No check performed
			LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 [j]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
				REF_CHK = 0	No check performed
	No		No protection information available to check.		

Table 28 — RDPROTECT field (Sheet 2 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
001b [b] [i]	Yes	Yes [e]	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
				GRD_CHK = 0	No check performed
			LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
				APP_CHK = 0	No check performed
			LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 [j]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
				REF_CHK = 0	No check performed
	No [a]	No protection information available to transmit to the data-in buffer or for checking			
010b [b] [i]	Yes	Yes [e]	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 [j]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
				REF_CHK = 0	No check performed
	No [a]	No protection information available to transmit to the data-in buffer or for checking			
011b [b] [i]	Yes	Yes [e]	LOGICAL BLOCK GUARD	No check performed	
			LOGICAL BLOCK APPLICATION TAG	No check performed	
			LOGICAL BLOCK REFERENCE TAG	No check performed	
	No [a]	No protection information available to transmit to the data-in buffer or for checking			

Table 28 — RDPROTECT field (Sheet 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
100b [b] [i]	Yes	Yes [e]	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
				GRD_CHK = 0	No check performed
			LOGICAL BLOCK APPLICATION TAG	No check performed	
			LOGICAL BLOCK REFERENCE TAG	No check performed	
	No [a]	No protection information available to transmit to the data-in buffer or for checking			
101b - 111b	Reserved				

[a] A read operation to a logical unit that supports protection information (see 4.16) and has not been formatted with protection information shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information the requested command should be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[c] The device server shall check the logical block application tag if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the READ (32) command (see 5.9) is used and the ATO bit is set to one in the Control mode page (see SPC-3), this knowledge is acquired from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge may be acquired by a method not defined by this standard.

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

[e] Transmit protection information to the data-in buffer.

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

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

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

[i] If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11), the device server may process the command. If the RTO_EN bit is set to one, READ (10) commands, READ (12) commands, and READ (16) commands with the RDPROTECT field set to 000b may be processed by the device server. If the RTO_EN bit is set to one, the device server shall terminate READ (10) commands, READ (12) commands, and READ (16) commands with the RDPROTECT field not set to 000b with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE.

[j] If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11), the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If the RTO_EN bit is set to one (i.e., the command is a READ (32) command), the device server checks the logical block reference tag based on the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB (see 4.16.2).

A disable page out (DPO) bit set to zero specifies that the retention priority shall be determined by the RETENTION PRIORITY fields in the Caching mode page (see 6.3.3). A DPO bit set to one specifies that the device

server shall assign the logical blocks accessed by this command the lowest retention priority for being fetched into or retained by the cache. A DPO bit set to one overrides any retention priority specified in the Caching mode page. All other aspects of the algorithm implementing the cache replacement strategy are not defined in this standard.

NOTE 1 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, the application client is specifying that the logical blocks accessed by the command are not likely to be accessed again in the near future and should not be put in the cache nor retained by the cache. If the DPO bit is set to zero, the application client is specifying that the logical blocks accessed by this command are likely to be accessed again in the near future.

The force unit access (FUA) and force unit access non-volatile cache (FUA_NV) bits are defined in table 29.

Table 29 — Force unit access for read operations

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

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be read and transferred to the data-in buffer, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A TRANSFER LENGTH field set to zero specifies that no logical blocks shall be read. This condition shall not be considered an error. Any other value specifies the number of logical blocks that shall be read. If the logical block address plus the transfer length exceeds the capacity of the medium, 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 in the Block Limits VPD page (see 6.4.2).

NOTE 2 For the READ (6) command, a TRANSFER LENGTH field set to zero specifies that 256 logical blocks are read.

5.7 READ (12) command

The READ (12) command (see table 30) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is

formatted with protection information enabled, protection information. Each logical block transferred includes user data and may include protection information, based on the RDPROTECT field and the medium format.

Table 30 — READ (12) command

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

See the READ (10) command (see 5.6) for the definitions of the fields in this command.

5.8 READ (16) command

The READ (16) command (see table 31) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes user data and may include protection information, based on the RDPROTECT field and the medium format.

Table 31 — READ (16) command

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

See the READ (10) command (see 5.6) for the definitions of the fields in this command.

5.9 READ (32) command

The READ (32) command (see table 32) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes user data and may include protection information, based on the RDPROTECT field and the medium format.

If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE. If the RTO_EN bit is set to one, the device server may process the command.

Table 32 — READ (32) command

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

See the READ (10) command (see 5.6) for the definitions of the GROUP NUMBER field, the RDPROTECT field, the DPO bit, the FUA bit, the FUA_NV bit, the LOGICAL BLOCK ADDRESS field and the TRANSFER LENGTH field.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 28 in 5.6), 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 4.16.2).

If the ATO bit is set to one in the Control mode page (see SPC-3) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 28 in 5.6), 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 the 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 the protection information.

If the ATO bit is set to one in the Control mode page (see SPC-3) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 28 in 5.6), or if the ATO bit is set to zero, the LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored.

5.10 READ CAPACITY (10) command

5.10.1 READ CAPACITY (10) overview

The READ CAPACITY (10) command (see table 33) requests that the device server transfer 8 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 had a HEAD OF QUEUE task attribute (see 4.11). If the logical unit supports protection information (see 4.16), the application client should use the READ CAPACITY (16) command instead of the READ CAPACITY (10) command.

Table 33 — READ CAPACITY (10) command

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

See the PRE-FETCH (10) command (see 5.3) for the definition of the LOGICAL BLOCK ADDRESS field.

The LOGICAL BLOCK ADDRESS field shall be set to zero if the PMI bit is set to zero. If the PMI bit is set to zero and the LOGICAL BLOCK ADDRESS field is not set to zero, the device server shall 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 partial medium indicator (PMI) bit set to zero specifies that the device server return information on the last logical block on the direct-access block device.

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

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

5.10.2 READ CAPACITY (10) parameter data

The READ CAPACITY (10) parameter data is defined in table 34. Any time the READ CAPACITY (10) parameter data changes, the device server should establish a unit attention condition as described in 4.6.

Table 34 — READ CAPACITY (10) parameter data

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

If the number of logical blocks exceeds the maximum value that is able to be specified in the RETURNED LOGICAL BLOCK ADDRESS field, the device server shall set the RETURNED LOGICAL BLOCK ADDRESS field to FFFFFFFFh. The application client should then issue a READ CAPACITY (16) command (see 5.11) to retrieve the READ CAPACITY (16) parameter data.

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

a) the LBA of the last logical block on the direct-access block device or
b) FFFFFFFFh.

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

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

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

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

5.11 READ CAPACITY (16) command

5.11.1 READ CAPACITY (16) command overview

The READ CAPACITY (16) command (see table 35) 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 4.16) and optional otherwise. This command is implemented as a service action of the SERVICE ACTION IN operation code (see A.2). This command may be processed as if it had a HEAD OF QUEUE task attribute (see 4.11).

Table 35 — READ CAPACITY (16) command

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

See the READ CAPACITY (10) command (see 5.10) for definitions of the LOGICAL BLOCK ADDRESS field and the PMI bit.

The ALLOCATION LENGTH field specifies the maximum number of bytes that the application client has allocated for returned parameter data. An allocation length of zero indicates that no data shall be transferred. This condition shall not be considered as an error. The device server shall terminate transfers to the data-in buffer when the number of bytes specified by the ALLOCATION LENGTH field have been transferred or when all available data has been transferred, whichever is less. The contents of the parameter data shall not be altered to reflect the truncation, if any, that results from an insufficient allocation length.

5.11.2 READ CAPACITY (16) parameter data

The READ CAPACITY (16) parameter data is defined in table 36. Any time the READ CAPACITY (16) parameter data changes, the device server should establish a unit attention condition as described in 4.6.

Table 36 — READ CAPACITY (16) parameter data

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

The RETURNED LOGICAL BLOCK ADDRESS field and BLOCK LENGTH IN BYTES field of the READ CAPACITY (16) parameter data are the same as in the READ CAPACITY (10) parameter data (see 5.10). The maximum value that shall be returned in the RETURNED LOGICAL BLOCK ADDRESS field is FFFFFFFF_FFFFFFFEh.

A reference tag own enable (RTO_EN) bit set to one indicates that application client ownership of the LOGICAL BLOCK REFERENCE TAG field in protection information is enabled (i.e., the medium was formatted with protection information (see 4.16) enabled and the RTO_REQ bit was set to one). An RTO_EN bit set to zero indicates that application client ownership of the LOGICAL BLOCK REFERENCE TAG field in protection information is disabled.

A PROT_EN bit set to one indicates that the medium was formatted with protection information (see 4.16) enabled. A PROT_EN bit set to zero indicates that the medium was not formatted with protection information enabled.

5.12 READ DEFECT DATA (10) command

5.12.1 READ DEFECT DATA (10) command overview

The READ DEFECT DATA (10) command (see table 37) requests that the device server transfer the medium defect data to the data-in buffer.

Table 37 — READ DEFECT DATA (10) command

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

If the device server is unable to access the medium defect data, it shall terminate the command with CHECK CONDITION status. The sense key shall be set to either MEDIUM ERROR, if a medium error occurred, or NO

SENSE, if medium defect data does not exist. The additional sense code shall be set to DEFECT LIST NOT FOUND.

NOTE Some device servers may not be able to return medium defect data until after a FORMAT UNIT command (see 5.2) has been completed successfully.

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

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

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

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

The DEFECT LIST FORMAT field specifies the preferred format for the defect list. This field is intended for those device servers capable of returning more than one format, as defined in the FORMAT UNIT command (see 5.2.2.4). A device server unable to return the requested format shall return the defect list in its default format and indicate that format in the DEFECT LIST FORMAT field in the defect list header (see table 38).

If the requested defect list format and the returned defect list format are not the same, the device server shall transfer the defect data and then terminate the command with CHECK CONDITION status 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 the READ CAPACITY (16) command (see 5.11). The application client is responsible for comparing the allocation length requested in the CDB with the defect list length returned in the parameter data to determine whether a partial list was received. If the number of address descriptors the device server has to report exceeds the maximum value that is able to be specified in the ALLOCATION LENGTH field, the device server shall transfer no data and return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

5.12.2 READ DEFECT DATA (10) parameter data

The READ DEFECT DATA (10) parameter data (see table 38) contains a four-byte header, followed by zero or more address descriptors.

Table 38 — READ DEFECT DATA (10) parameter data

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

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

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

The DEFECT LIST FORMAT field indicates the format of the address descriptors returned by the device server. This field is defined in the FORMAT UNIT command (see 5.2.2.4).

If the device server returns short block format address descriptors (see 5.2.2.4.2) or long block format address descriptors (see 5.2.2.4.3), the address descriptors contain vendor-specific values.

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

If the device server returns physical sector format address descriptors (see 5.2.2.4.5), it may or may not include defects in parts of the medium not accessible to the application client. If the device server returns bytes from index format address descriptors (see 5.2.2.4.4), it shall return a complete list of the defects. A complete list of the defects may include defects in areas not within the capacity returned in the READ CAPACITY command.

The DEFECT LIST LENGTH field indicates the length in bytes of the address descriptors that follow. The DEFECT LIST LENGTH is equal to four or eight times the number of the address descriptors, depending on the format of the returned address descriptors (see 5.2.2.4).

The address descriptors may or may not be sent in ascending order.

5.13 READ DEFECT DATA (12) command

5.13.1 READ DEFECT DATA (12) command overview

The READ DEFECT DATA (12) command (see table 39) requests that the device server transfer the medium defect data to the data-in buffer.

Table 39 — READ DEFECT DATA (12) command

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

See the READ DEFECT DATA (10) command (see 5.11) for the definitions of the fields in this command.

NOTE The application client may determine the length of the defect list by sending the READ DEFECT DATA (12) command with an ALLOCATION LENGTH field set to eight. The device server returns the defect list header that contains the length of the defect list.

5.13.2 READ DEFECT DATA (12) parameter data

The READ DEFECT DATA (12) parameter data (see table 40) contains an eight byte header, followed by zero or more address descriptors.

Table 40 — READ DEFECT DATA (12) parameter data

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

See the READ DEFECT DATA (10) command (see 5.12) for the definitions of the fields in the defect list.

5.14 READ LONG (10) command

The READ LONG (10) command (see table 41) requests that the device server transfer data from a single logical block to the data-in buffer. The data transferred during the READ LONG (10) command is vendor-specific, but shall include the following items recorded on the medium:

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

If a cache contains a more recent version of a logical block, the device server shall write the logical block to the medium before reading it. The values in the Read-Write Error Recovery mode page (see 6.3.4) do not apply to this command. The device server may perform retries while processing this command.

Table 41 — READ LONG (10) command

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

See the PRE-FETCH (10) command (see 5.3) for the definition of the LOGICAL BLOCK ADDRESS field.

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

bytes to be at the end of the user data or protection information, if any; however, the ECC bytes should be in the same order as they are on the medium.

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

The BYTE TRANSFER LENGTH field specifies the number of bytes of data that shall be read from the specified logical block and transferred to the data-in buffer. If the BYTE TRANSFER LENGTH field is not set to zero and does not match the available data length, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. In the sense data (see 4.13 and SPC-3), the VALID and ILI bits shall each be set to one and the INFORMATION field shall be set to the difference (i.e., residue) of the requested byte transfer length minus the actual available data length in bytes. Negative values shall be indicated by two's complement notation.

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

5.15 READ LONG (16) command

The READ LONG (16) command (see table 42) requests that the device server transfer data from a single logical block to the data-in buffer. The data transferred during the READ LONG (16) command is vendor-specific, but shall include the following items recorded on the medium:

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

If a cache contains a more recent version of a logical block, the device server shall write the logical block to the medium before reading it. The values in the Read-Write Error Recovery mode page (see 6.3.4) do not apply to this command. The device server may perform retries while processing this command. This command is implemented as a service action of the SERVICE ACTION IN operation code (see A.2).

Table 42 — READ LONG (16) command

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

See the READ LONG (10) command (see 5.14) for the definitions of the fields in this command.

5.16 REASSIGN BLOCKS command

5.16.1 REASSIGN BLOCKS command overview

The REASSIGN BLOCKS command (see table 43) requests that the device server reassign defective logical blocks to another area on the medium set aside for this purpose. The device server should also record the location of the defective logical blocks in the GLIST, if supported. This command shall not alter the contents of the PLIST (see 4.8).

The parameter list provided in the data-out buffer contains a defective 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 in the defective LBA list. More than one physical block may be relocated by each LBA. If the device server is able to recover user data and protection information, if any, from the original logical block, it shall write the recovered user data and any protection information to the reassigned logical block. If the device server is unable to recover user data and protection information, if any, it shall write vendor-specific data as the user data and shall write a default value of FFFFFFFF_FFFFFFFFh as the protection information, if enabled. The data in all other logical blocks on the medium shall be preserved.

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

Table 43 — REASSIGN BLOCKS command

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

A long LBA (LONGLBA) bit set to zero specifies that the REASSIGN BLOCKS defective LBA list contains four byte LBAs. A LONGLBA bit set to one specifies that the REASSIGN BLOCKS defective LBA list contains eight byte LBAs.

5.16.2 REASSIGN BLOCKS parameter list

The REASSIGN BLOCKS parameter list (see table 44) contains a four-byte parameter list header followed by a defective LBA list containing one or more LBAs.

Table 44 — REASSIGN BLOCKS parameter list

Byte\Bit	7	6	5	4	3	2	1	0
0	Parameter list header (see table 45 or table 46)							
3								
4	DEFECTIVE LBA LIST (if any)							
n								

If LONGLIST is set to zero, the parameter list header is defined in table 45.

Table 45 — REASSIGN BLOCKS short parameter list header

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

If LONGLIST is set to one, the parameter list header is defined in table 46.

Table 46 — REASSIGN BLOCKS long parameter list header

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

The DEFECT LIST LENGTH field indicates the total length in bytes of the DEFECTIVE LBA LIST field. The DEFECT LIST LENGTH field does not include the parameter list header length and is equal to either

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

The DEFECTIVE LBA LIST field contains a list of defective LBAs. Each LBA is a four-byte field if the LONGLBA bit is set to zero or an eight-byte field if the LONGLBA bit is set to one. The LBAs shall be in ascending order.

If the direct-access block device has insufficient capacity to reassign all of the specified logical blocks, 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 successfully complete a REASSIGN BLOCKS command, the device server shall terminate the command with CHECK CONDITION status with the appropriate sense data (see 4.13 and SPC-3). The first LBA not reassigned shall be returned in the COMMAND-SPECIFIC INFORMATION field of the sense data. If information about the first LBA not reassigned is not available, or if all the defects have been reassigned, the COMMAND-SPECIFIC INFORMATION field shall be set to FFFFFFFFh if fixed format sense data is being used or FFFFFFFF_FFFFFFFFh if descriptor format sense data is being used.

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

NOTE If the REASSIGN BLOCKS command returns CHECK CONDITION status and the sense data COMMAND-SPECIFIC INFORMATION field contains a valid LBA, the application client should remove all LBAs from the defective LBA list prior to the one returned in the COMMAND-SPECIFIC INFORMATION field. If the sense key is MEDIUM ERROR and the INFORMATION field contains the valid LBA, the application client should insert that new defective LBA into the defective LBA list and reissue the REASSIGN BLOCKS command with the new defective 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 defective LBA list.

5.17 START STOP UNIT command

The START STOP UNIT command (see table 47) requests that the device server change the power condition of the logical unit (see 4.15) 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.

Logical units that contain cache shall write all cached logical blocks to the medium (e.g., as they would do in response to a SYNCHRONIZE CACHE command (see 5.18 and 5.19) with the SYNC_NV bit set to zero, the LOGICAL BLOCK ADDRESS field set to zero and the NUMBER OF BLOCKS field set to zero) prior to entering into any

power condition that prevents accessing the medium (e.g., before the rotating media spindle motor is stopped during transition to the stopped power condition).

Table 47 — START STOP UNIT command

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

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 POWER CONDITION field is used to specify that the logical unit be placed into a power condition or to adjust a timer as defined in table 48. If this field is supported and is set to a value other than 0h, then the START and LOEJ bits shall be ignored.

Table 48 — POWER CONDITION field

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

If the START STOP UNIT command is received with the POWER CONDITION field set to ACTIVE, IDLE or STANDBY, then

a) the logical unit shall transition to the specified power condition,
b) the logical unit shall change power conditions only after receipt of another START STOP UNIT command or a logical unit reset,
c) the device server shall disable the idle condition timer if it is active (see SPC-3) and disable the standby condition timer if it is active (see SPC-3) until another START STOP UNIT command is received that returns control of the power condition to the logical unit, or a logical unit reset occurs.

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

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

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

It is not an error to specify that the logical unit transition to its current power condition.

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

If the START bit is set to zero, then the logical unit shall transition to the stopped power condition, disable the idle condition timer if it is active (see SPC-3) and disable the standby condition timer if it is active (see SPC-3). If the START bit set is to one, then the logical unit shall transition to the active power condition, enable the idle condition timer if it is active and enable the standby condition timer if it is active.

5.18 SYNCHRONIZE CACHE (10) command

The SYNCHRONIZE CACHE (10) command (see table 49) requests that the device server ensure that the specified logical blocks have their most recent data values recorded in non-volatile cache and/or on the medium, based on the SYNC_NV bit. Logical blocks include user data and, if the medium is formatted with protection information enabled, protection information. Logical blocks may or may not be removed from volatile cache and non-volatile cache as a result of the synchronize cache operation.

Table 49 — SYNCHRONIZE CACHE (10) command

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

See the PRE-FETCH (10) command (see 5.3) for the definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command (see 5.3) and 4.17 for the definition of the GROUP NUMBER field.

The SYNC_NV bit (see table 50) specifies whether the device server is required to synchronize volatile and non-volatile caches.

Table 50 — SYNC_NV bit

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

An immediate (IMMED) bit set to zero specifies that the device server shall not return status until the operation has been completed. An IMMED bit set to one specifies that the device server shall return status as soon as the CDB has been validated. If the IMMED bit is set to one and the device server does not support the IMMED bit, the 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 NUMBER OF BLOCKS field specifies the number of logical blocks that shall be synchronized, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A NUMBER OF BLOCKS field set to zero specifies that all logical blocks starting with the one specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium shall be synchronized. If the logical block address plus the number of blocks exceeds the capacity of the medium, 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.

5.19 SYNCHRONIZE CACHE (16) command

The SYNCHRONIZE CACHE (16) command (see table 51) requests that the device server ensure that the specified logical blocks have their most recent data values recorded in non-volatile cache and/or on the medium, based on the SYNC_NV bit. Logical blocks include user data and, if the medium is formatted with protection information enabled, protection information. Logical blocks may or may not be removed from volatile cache and non-volatile cache as a result of the synchronize cache operation.

Table 51 — SYNCHRONIZE CACHE (16) command

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

See the SYNCHRONIZE CACHE (10) command (see 5.18) for the definitions of the fields in this command.

5.20 VERIFY (10) command

The VERIFY (10) command (see table 52) requests that the device server verify the specified logical block(s) on the medium. Each logical block includes user data and may include protection information, based on the VRPROTECT field and the medium format.

If the RTO_EN bit is set to one in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE.

NOTE This command description is referenced by the VERIFY (32) command, which is terminated when the RTO_EN bit is set to zero rather than one.

Table 52 — VERIFY (10) command

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

Logical units that contain cache shall write referenced cached logical blocks to the medium for the logical unit (e.g., as they would do in response to a SYNCHRONIZE CACHE command (see 5.18 and 5.19) with the SYNC_NV bit set to zero, the LOGICAL BLOCK ADDRESS field set to the value of the VERIFY command's LOGICAL BLOCK ADDRESS field, and the NUMBER OF BLOCKS field set to the value of the VERIFY command's VERIFICATION LENGTH field).

See the READ (10) command (see 5.6) for the definition of the DPO bit. See the PRE-FETCH (10) command (see 5.3) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.3) and 4.17 for the definition of the GROUP NUMBER field.

If the Verify Error Recovery mode page (see 6.3.5) is implemented, then the current settings in that page specify the verification criteria. If the Verify Error Recovery mode page is not implemented, then the verification criteria is vendor-specific.

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

a) perform a medium verification with no data comparison and not transfer any data from the data-out buffer and
b) check protection information read from the medium based on the VRPROTECT field as described in table 53.

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

a) perform a byte-by-byte comparison of user data read from the medium and user data transferred from the data-out buffer,
b) check protection information read from the medium based on the VRPROTECT field as described in table 54,
c) check protection transferred from the data-out buffer based on the VRPROTECT field as described in table 55 and
d) perform a byte-by-byte comparison of protection information read from the medium and transferred from the data-out buffer based on the VRPROTECT field as described in table 56.

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

If a byte-by-byte comparison is unsuccessful for any reason, the device server shall 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 specified by the LOGICAL BLOCK ADDRESS field. If the BYTCHK bit is set to one, the VERIFICATION LENGTH field also specifies the number of logical blocks that the device server shall transfer from the data-out buffer. A VERIFICATION LENGTH field set to zero specifies that no logical blocks shall be verified. This condition shall not be considered as an error. Any other value specifies the number of logical blocks that shall be verified. If the logical block address plus the verification length exceeds the capacity of the medium, the device server shall 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 in the Block Limits VPD page (see 6.4.2).

If the BYTCHK bit is set to zero, the device server shall check the protection information read from the medium based on the VRPROTECT field as described in table 53.

Table 53 — VRPROTECT field with BYTCHK set to zero - checking protection information read from the medium (Sheet 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	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 [h]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	No protection information on the medium to check. Only user data is checked.		
001b [b]	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 [c]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 [h]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	Error condition [a]		

Table 53 — VRPROTECT field with BYTCHK set to zero - checking protection information read from the medium (Sheet 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
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]		
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]		

Table 53 — VRPROTECT field with BYTCHK set to zero - checking protection information read from the medium (Sheet 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
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]		
101b-111b	Reserved			

[a] A verify operation to a logical unit that supports protection information (see 4.16) and has not been formatted with protection information shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information the requested command should be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[c] The device server shall check the logical block application tag if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the VERIFY (32) command (see 5.23) is used and the ATO bit is set to one in the Control mode page (see SPC-3), this knowledge is acquired from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge may be obtained by a method not defined by this standard.

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

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

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

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

[h] If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11)(i.e., the command is a VERIFY (10) command, a VERIFY (12) command, or a VERIFY (16) command), the device server shall check the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If the RTO_EN bit is set to one (i.e., the command is a VERIFY (32) command), the device server shall check the logical block reference tag based on the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB (see 4.16.2).

If the BYTCHK bit is set to one, the device server shall check the protection information read from the medium based on the VRPROTECT field as described in table 54.

Table 54 — VRPROTECT field with BYTCHK set to one – Checking protection information read from the medium (Sheet 1 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
000b	Yes	LOGICAL BLOCK GUARD	GRD_CHK = 1	LOGICAL BLOCK GUARD CHECK FAILED
			GRD_CHK = 0	No check performed
		LOGICAL BLOCK APPLICATION TAG	APP_CHK = 1 [c] [g]	LOGICAL BLOCK APPLICATION TAG CHECK FAILED
			APP_CHK = 0	No check performed
		LOGICAL BLOCK REFERENCE TAG	REF_CHK = 1 [h]	LOGICAL BLOCK REFERENCE TAG CHECK FAILED
			REF_CHK = 0	No check performed
	No	No protection information on the medium available to check		
001b 010b 011b 100b [b]	Yes	LOGICAL BLOCK GUARD	No check performed	
		LOGICAL BLOCK APPLICATION TAG	No check performed	
		LOGICAL BLOCK REFERENCE TAG	No check performed	
	No	Error condition [a]		

Table 54 — VRPROTECT field with BYTCHK set to one – Checking protection information read from the medium (Sheet 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
101b-111b	Reserved			

[a] A verify operation to a logical unit that supports protection information (see 4.16) and has not been formatted with protection information shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information the requested command should be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[c] The device server shall check the logical block application tag if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the VERIFY (32) command (see 5.23) is used and the ATO bit is set to one in the Control mode page (see SPC-3), this knowledge is acquired from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge may be obtained by a method not defined by this standard.

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

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

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

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

[h] If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11)(i.e., the command is a VERIFY (10) command, a VERIFY (12) command, or a VERIFY (16) command), the device server shall check the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If the RTO_EN bit is set to one (i.e., the command is a VERIFY (32) command), the device server shall check the logical block reference tag based on the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB (see 4.16.2).

If the BYTCHK bit is set to one, the device server shall check the protection information transferred from the data-out buffer based on the VRPROTECT field as described in table 55.

Table 55 — VRPROTECT field with BYTCHK set to one - checking protection information transferred from the data-out buffer (Sheet 1 of 3)

Code	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails [d] [e], additional sense code
000b	Yes	No protection information received from application client to check		
	No	No protection information received from application client to check		

Table 55 — VRPROTECT field with BYTCHK set to one - checking protection information transferred from the data-out buffer (Sheet 2 of 3)

Code	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails [d] [e], additional sense code
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 [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]		

Table 55 — VRPROTECT field with BYTCHK set to one - checking protection information transferred from the data-out buffer (Sheet 3 of 3)

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

[a] A verify operation to a logical unit that supports protection information (see 4.16) and has not been formatted with protection information shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information the requested command should be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[c] The device server may check the logical block application tag if the ATO bit is set to one in the Control mode page (see SPC-3) and if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the VERIFY (32) command (see 5.23) is used, this knowledge is obtained from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge is obtained by a method not defined by this standard.

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

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

[f] If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11)(i.e., the command is a VERIFY (10) command, a VERIFY (12) command or a VERIFY (16) command), the device server shall check the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If the RTO_EN bit is set to one (i.e., the command is a VERIFY (32) command), the device server shall check the logical block reference tag based on the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB (see 4.16.2).

If the BYTCHK bit is set to one, the device server shall perform a byte-by-byte comparison of protection information transferred from the data-out buffer with protection information read from the medium based on the VRPROTECT field as described in table 56.

Table 56 — VRPROTECT field with BYTCHK set to one – Byte-by-byte comparison requirements (Sheet 1 of 2)

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

Table 56 — VRPROTECT field with BYTCHK set to one – Byte-by-byte comparison requirements (Sheet 2 of 2)

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

[a] A verify operation to a logical unit that supports protection information (see 4.16) and has not been formatted with protection information shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
[b] If the logical unit does not support protection information the requested command should be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
[c] If an error is reported, the sense key shall be set to MISCOMPARE.
[d] If multiple errors occur, the selection of which error to report is not defined by this standard.
[e] If the ATO bit is set to one in the Control mode page (see SPC-3), the logical block application tag shall not be modified by a device server.
[f] If the ATO bit is set to zero in the Control mode page (see SPC-3), the logical block application tag may be modified by a device server.

5.21 VERIFY (12) command

The VERIFY (12) command (see table 57) requests that the device server verify the specified logical block(s) on the medium. Each logical block includes user data and may include protection information, based on the VRPROTECT field and the medium format.

If the RTO_EN bit is set to one in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE.

Table 57 — VERIFY (12) command

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

See the VERIFY (10) command (see 5.20) for the definitions of the fields in this command.

5.22 VERIFY (16) command

The VERIFY (16) command (see table 58) requests that the device server verify the specified logical block(s) on the medium. Each logical block includes user data and may include protection information, based on the VRPROTECT field and the medium format.

If the RTO_EN bit is set to one in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE.

Table 58 — VERIFY (16) command

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

See the VERIFY (10) command (see 5.20) for the definitions of the fields in this command.

5.23 VERIFY (32) command

The VERIFY (32) command (see table 59) requests that the device server verify the specified logical block(s) on the medium. Each logical block includes user data and may include protection information, based on the VRPROTECT field and the medium format.

If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE.

Table 59 — VERIFY (32) command

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

See the VERIFY (10) command (see 5.20) for the definitions of the GROUP NUMBER field, VRPROTECT field, DPO bit, BYTCHK bit, LOGICAL BLOCK ADDRESS field, and VERIFICATION LENGTH field.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 53, table 54, table 55, and table 56 in 5.20), 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 4.16.2).

If the ATO bit is set to one in the Control mode page (see SPC-3) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 53, table 54, table 55 and table 56 in 5.20), 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 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 the protection information.

If the ATO bit is set to one in the Control mode page (see SPC-3) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 53, table 54, table 55 and table 56 in 5.20), or if the ATO bit is set to zero, the LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored.

5.24 WRITE (6) command

The WRITE (6) command (see table 60) requests that the device server transfer the specified logical block(s) from the data-out buffer and write them. Each logical block transferred includes user data but does not include protection information. Each logical block written includes user data and, if the medium is formatted with protection information enabled, protection information.

Table 60 — WRITE (6) command

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

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

See the PRE-FETCH (10) command (see 5.3) for the definition of the LOGICAL BLOCK ADDRESS 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 specified by the LOGICAL BLOCK ADDRESS field. A TRANSFER LENGTH field set to zero specifies that 256 logical blocks shall be written. Any other value specifies the number of logical blocks that shall be written. If the logical block address plus the transfer length exceeds the capacity of the medium, 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 in the Block Limits VPD page (see 6.4.2).

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

If a WRITE (6) command is received after protection information is enabled the device server shall set the protection information (see 4.16) as follows as it writes each logical block to the medium:

a) the LOGICAL BLOCK GUARD field set to a properly generated CRC (see 4.16.3);
b) the LOGICAL BLOCK REFERENCE TAG field set to:
   A) the least significant four bytes of the LBA, if the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11); or
   B) FFFFFFFFh, if the RTO_EN bit is set to one;

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

5.25 WRITE (10) command

The WRITE (10) command (see table 61) requests that the device server transfer the specified logical block(s) from the data-out buffer and write them. Each logical block transferred includes user data and may include protection information, based on the WRPROTECT field and the medium format. Each logical block

written includes user data and, if the medium is formatted with protection information enabled, protection information.

Table 61 — WRITE (10) command

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

See the READ (10) command (see 5.6) for the definition of the DPO bit. See the PRE-FETCH (10) command (see 5.3) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.3) and 4.17 for the definition of the GROUP NUMBER field.

The device server shall check the protection information transferred from the data-out buffer based on the WRPROTECT field as described in table 62.

Table 62 — WRPROTECT field (Sheet 1 of 3)

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

Table 62 — WRPROTECT **field (Sheet 2 of 3)**

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

Table 62 — WRPROTECT **field (Sheet 3 of 3)**

Code	Logical unit formatted with protection information	Field in protection information	Device server check	If check fails [d] [i], additional sense code
101b - 111b	Reserved			

[a] A write operation to a logical unit that supports protection information (see 4.16) and has not been formatted with protection information shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information the requested command should be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[c] The device server may check the logical block application tag if the ATO bit is set to one in the Control mode page (see SPC-3) and if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the WRITE (32) command (see 5.28) is used, this knowledge is obtained from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge is obtained by a method not defined by this standard.

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

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

[f] The device server shall write a properly generated CRC (see 4.16.3.2) into each LOGICAL BLOCK GUARD field.

[g] If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11), the device server shall write the least significant four bytes of each LBA into the LOGICAL BLOCK REFERENCE TAG field of each of the written logical blocks. If the RTO_EN bit is set to one, the device server shall write a value of FFFFFFFFh into the LOGICAL BLOCK REFERENCE TAG field of each of the written logical blocks.

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

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

[j] If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11), the device server may process the command. If the RTO_EN bit is set to one, WRITE (10) commands, WRITE (12) commands and WRITE (16) commands with the WRPROTECT field set to 000b may be processed by the device server. If the RTO_EN bit is set to one, the device server shall terminate WRITE (10) commands, WRITE (12) commands and WRITE (16) commands with the WRPROTECT field not set to 000b with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE.

[k] If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11), the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If the RTO_EN bit is set to one (i.e., the command is a WRITE (32) command), the device server checks the logical block reference tag based on the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB (see 4.16.2).

The force unit access (FUA) and force unit access non-volatile cache (FUA_NV) bits are defined in table 63.

Table 63 — Force unit access for write operations

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

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

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred from the data-out buffer and written, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A TRANSFER LENGTH 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. If the logical block address plus the transfer length exceeds the capacity of the medium, 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 in the Block Limits VPD page (see 6.4.2).

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

5.26 WRITE (12) command

The WRITE (12) command (see table 64) requests that the device server transfer the specified logical block(s) from the data-out buffer and write them. Each logical block transferred includes user data and may include protection information, based on the WRPROTECT field and the medium format. Each logical block written includes user data and, if the medium is formatted with protection information enabled, protection information.

Table 64 — WRITE (12) command

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

See the WRITE (10) command (see 5.25) for the definitions of the fields in this command.

5.27 WRITE (16) command

The WRITE (16) command (see table 65) requests that the device server transfer the specified logical block(s) from the data-out buffer and write them. Each logical block transferred includes user data and may include protection information, based on the WRPROTECT field and the medium format. Each logical block written includes user data and, if the medium is formatted with protection information enabled, protection information.

Table 65 — WRITE (16) command

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

See the WRITE (10) command (see 5.25) for the definitions of the fields in this command.

5.28 WRITE (32) command

The WRITE (32) command (see table 66) requests that the device server transfer the specified logical block(s) from the data-out buffer and write them. Each logical block transferred includes user data and may include protection information, based on the WRPROTECT field and the medium format. Each logical block written includes user data and, if the medium is formatted with protection information enabled, protection information.

If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11), the device server shall terminate the WRITE (32) command with CHECK CONDITION status with the sense key set to ILLEGAL

REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE. If the RTO_EN bit is set to one, the device server may process the command.

Table 66 — WRITE (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
5								
6	Reserved			GROUP NUMBER				
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)			SERVICE ACTION (000Bh)				
9								(LSB)
10	WRPROTECT			DPO	FUA	Reserved	FUA_NV	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)

See the WRITE (10) command (see 5.25) for the definitions of the GROUP NUMBER field, the WRPROTECT field, the DPO bit, the FUA bit, the FUA_NV bit, the LOGICAL BLOCK ADDRESS field and the TRANSFER LENGTH field.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 62 in 5.25), 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 4.16.2).

If the ATO bit is set to one in the Control mode page (see SPC-3) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 62 in 5.25), 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 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 the protection information.

If the ATO bit is set to one in the Control mode page (see SPC-3) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 62 in 5.25), or if the ATO bit is set to zero, the LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored.

5.29 WRITE AND VERIFY (10) command

The WRITE AND VERIFY (10) command (see table 67) requests that the device server transfer the specified logical block(s) from the data-out buffer, write them to the medium and then verify that they are correctly written. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format. The logical blocks are only transferred once from the data-out buffer to the device server.

If the RTO_EN bit is set to one in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE.

Table 67 — WRITE AND VERIFY (10) command

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

See the PRE-FETCH (10) command (see 5.3) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.3) and 4.17 for the definition of the GROUP NUMBER field. See the WRITE (10) command (see 5.25) for the definitions of the TRANSFER LENGTH field and the WRPROTECT field. See the READ (10) command (see 5.6) for the definition of the DPO bit.

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

A byte check (BYTCHK) bit set to zero specifies that, after writing, the device server perform a medium verification with no data comparison. A BYTCHK bit set to one specifies that, after writing, the device server perform a byte-by-byte comparison of data written on the medium with the data just written. If the comparison is unsuccessful for any reason, the device server shall 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.

5.30 WRITE AND VERIFY (12) command

The WRITE AND VERIFY (12) command (see table 68) requests that the device server transfer the specified logical block(s) from the data-out buffer, write them to the medium and then verify that they are correctly written. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format. The logical blocks are only transferred once from the data-out buffer to the device server.

If the RTO_EN bit is set to one in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE.

Table 68 — 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	Restricted for MMC-4	Reserved		GROUP NUMBER				
11	CONTROL							

See the WRITE AND VERIFY (10) command (see 5.29) for the definitions of the fields in this command.

5.31 WRITE AND VERIFY (16) command

The WRITE AND VERIFY (16) command (see table 69) requests that the device server transfer the specified logical block(s) from the data-out buffer, write them to the medium, and then verify that they are correctly written. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format. The logical blocks are only transferred once from the data-out buffer to the device server.

If the RTO_EN bit is set to one in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE

Table 69 — 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	Restricted for MMC-4	Reserved		GROUP NUMBER				
15	CONTROL							

See the WRITE AND VERIFY (10) command (see 5.29) for the definitions of the fields in this command.

5.32 WRITE AND VERIFY (32) command

The WRITE AND VERIFY (32) command (see table 70) requests that the device server transfer the specified logical block(s) from the data-out buffer, write them to the medium and then verify that they are correctly

written. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format. The logical blocks are only transferred once from the data-out buffer to the device server.

If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE. If the RTO_EN bit is set to one, the device server may process the command.

Table 70 — WRITE AND VERIFY (32) command

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

See the WRITE AND VERIFY (10) command (see 5.29) for the definitions of the GROUP NUMBER field, the WRPROTECT field, the DPO bit, the BYTCHK bit, the LOGICAL BLOCK ADDRESS field and the TRANSFER LENGTH field.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 62 in 5.25), 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 4.16.2).

If the ATO bit is set to one in the Control mode page (see SPC-3) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 62 in 5.25), 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 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 the protection information.

If the ATO bit is set to one in the Control mode page (see SPC-3) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 62 in 5.25), or if the ATO bit is set to zero, the LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored.

5.33 WRITE LONG (10) command

The WRITE LONG (10) command (see table 71) requests that the device server transfer data for a single logical block from the data-out buffer and write it to the medium.

The data written shall be the same length and shall be in the same order as the data returned by the READ LONG (10) command (see 5.14). The device server shall write the logical block to the medium and shall not return GOOD status until the logical block has actually been written on the medium.

Table 71 — WRITE LONG (10) command

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

See the PRE-FETCH (10) command (see 5.3) for the definition of the LOGICAL BLOCK ADDRESS field.

The BYTE TRANSFER LENGTH field specifies the number of bytes of data that the device server shall transfer from the data-out buffer and write to the specified logical block. If the BYTE TRANSFER LENGTH field is not set to zero and does not match the data length that the device server returns for a READ LONG command, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. In the sense data (see 4.13 and SPC-3), the ILI and VALID bits shall be set to one and the INFORMATION field shall be set to the difference (i.e., residue) of the requested length minus the actual length in bytes. Negative values shall be indicated by two's complement notation. A BYTE TRANSFER LENGTH field set to zero specifies that no bytes shall be written. This condition shall not be considered an error.

5.34 WRITE LONG (16) command

The WRITE LONG (16) command (see table 72) requests that the device server transfer data for a single logical block from the data-out buffer and write it to the medium. The data written shall be the same length and shall be in the same order as the data returned by the READ LONG (16) command (see 5.15). The device server shall write the logical block to the medium and shall not return GOOD status until the logical block has

actually been written on the medium. This command is implemented as a service action of the SERVICE ACTION OUT operation code (see A.2).

Table 72 — WRITE LONG (16) command

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

See the WRITE LONG (10) command (see 5.33) for the definitions of the fields in this command.

5.35 WRITE SAME (10) command

The WRITE SAME (10) command (see table 73) requests that the device server transfer a single logical block from the data-out buffer and write the contents of that logical block, with modifications based on the LBDATA bit and the PBDATA bit, to the specified range of logical block addresses. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format.

If the RTO_EN bit is set to one in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE.

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

Table 73 — WRITE SAME (10) command

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

See the WRITE (10) command (see 5.25) for the definitions of the WRPROTECT field. See the PRE-FETCH (10) command (see 5.3) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.3) and 4.17 for the definition of the GROUP NUMBER field.

Table 74 describes the LBDATA bit and the PBDATA bit.

Table 74 — LBDATA bit and PBDATA bit

LBDATA	PBDATA	Description
0	0	The device server shall write the single block of user data received from the data-out buffer to each logical block without modification. If the medium is formatted with protection information: a) the value in the LOGICAL BLOCK REFERENCE TAG field received in the single block of data from the data-out buffer shall be placed into the LOGICAL BLOCK REFERENCE TAG field of the first logical block written to the medium. Into each of the subsequent logical blocks, the device server shall place into the LOGICAL BLOCK REFERENCE TAG field the value of the previous logical block's LOGICAL BLOCK REFERENCE TAG field plus one; b) If the ATO bit is set to one in the Control mode page (see SPC-3), the logical block application tag received in the single block of data shall be placed in the LOGICAL BLOCK APPLICATION TAG field of each logical block. If the ATO bit is set to zero, the device server may write any value into the LOGICAL BLOCK APPLICATION TAG field of each logical block; and c) The value in the DATA BLOCK GUARD field received in the single block of data from the data-out buffer shall be placed in the DATA BLOCK GUARD field of each logical block.
0	1 [a]	The device server shall replace the first eight bytes of the block received from the data-out buffer to each physical sector with the physical address of the sector being written using the physical sector format (see 5.2.2.4.5).
1 [a]	0	The device server shall replace the first four bytes of the block received from the data-out buffer with the least significant four bytes of the LBA of the block being written, ending with the least significant byte (e.g., if the LBA is 77665544_33221100h, 33221100h is written with 33h written first and 00h written last).
1	1	The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

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

The NUMBER OF BLOCKS field specifies the number of contiguous logical blocks to be written, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A NUMBER OF BLOCKS field set to zero specifies that the device server write all the logical blocks starting with the one specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium. If the logical block address plus the number of blocks exceeds the capacity of the medium, 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.

5.36 WRITE SAME (16) command

The WRITE SAME (16) command (see table 75) requests that the device server transfer a single logical block from the data-out buffer and write the contents of that logical block, with modifications based on the LBDATA bit and the PBDATA bit, to the specified range of logical block addresses. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format.

If the RTO_EN bit is set to one in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE

Table 75 — WRITE SAME (16) command

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

See the WRITE SAME (10) command (see 5.35) for the definitions of the fields in this command.

5.37 WRITE SAME (32) command

The WRITE SAME (32) command (see table 76) requests that the device server transfer a single logical block from the data-out buffer and write the contents of that logical block, with modifications based on the LBDATA bit and the PBDATA bit, to the specified range of logical block addresses. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format.

If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE. If the RTO_EN bit is set to one, the device server may process the command.

Table 76 — WRITE SAME (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
5								
6	Reserved			GROUP NUMBER				
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)			SERVICE ACTION (000Dh)				
9								(LSB)
10	WRPROTECT			Reserved		PBDATA	LBDATA	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)			NUMBER OF BLOCKS				
31								(LSB)

See the WRITE SAME (10) command (see 5.35) for the definitions of the GROUP NUMBER field, the WRPROTECT field, the PBDATA bit, the LBDATA bit, the LOGICAL BLOCK ADDRESS field and the NUMBER OF BLOCKS field.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 62 in 5.25), 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 4.16.2).

If the ATO bit is set to one in the Control mode page (see SPC-3) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 62 in 5.25), 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 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 the protection information.

If the ATO bit is set to one in the Control mode page (see SPC-3) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 62 in 5.25), or if the ATO bit is set to zero, the LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored.

5.38 XDREAD (10) command

The XDREAD (10) command (see table 77) requests that the device transfer to the data-in buffer the XOR data generated by an XDWRITE command (see 5.40 and 5.41). XOR data includes user data and may include protection information, based on the XORPINFO bit and the medium format.

Table 77 — XDREAD (10) command

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

See the PRE-FETCH (10) command (see 5.3) and 4.17 for the definition of the GROUP NUMBER field.

If the XOR protection information (XORPINFO) bit is set to zero, the device server shall not check or transmit protection information.

If the XORPINFO bit is set to one, the device server supports protection information and the medium has been formatted with protection information, the device server shall transmit protection information but shall not check any of the protection information fields.

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

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

The XOR data transferred is identified by the LOGICAL BLOCK ADDRESS field and the TRANSFER LENGTH field. The LOGICAL BLOCK ADDRESS field and TRANSFER LENGTH field shall be the same as, or a subset of, those specified in a prior XDWRITE command. If a match is not found the 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 TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

5.39 XDREAD (32) command

The XDREAD (32) command (see table 78) requests that the device transfer to the data-in buffer the XOR data generated by an XDWRITE command (see 5.40 and 5.41). XOR data includes user data and may include protection information, based on the XORPINFO bit and the medium format.

Table 78 — XDREAD (32) command

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

See the XDREAD (10) command (see 5.38) and SPC-3 for the definitions of the fields in this command.

5.40 XDWRITE (10) command

The XDWRITE (10) command (see table 79) requests that the device server:

1) read the specified logical block(s);
2) transfer logical blocks from the data-out buffer;
3) perform an XOR operation with the logical blocks transferred from the data-out buffer and the logical blocks read, storing the resulting XOR data in a buffer; and
4) if the DISABLE WRITE bit is set to zero, write the logical blocks transferred from the data-out buffer.

Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format. The resulting XOR data shall be retained as specified in 4.14.4.

If the RTO_EN bit is set to one in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE.

Table 79 — XDWRITE (10) command

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

See the WRITE (10) command (see 5.25) for the definitions of the WRPROTECT field, the FUA bit, and the FUA_NV bit. See the READ (10) command (see 5.6) for the definition of the DPO bit. See the PRE-FETCH (10) command (see 5.3) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.3) and 4.17 for the definition of the GROUP NUMBER field.

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

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that read, transferred from the data-out buffer and XORed into a buffer, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. If the logical block address plus the transfer length exceeds the capacity of the medium, 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 in the Block Limits VPD page (see 6.4.2).

The resulting XOR data is retrieved by an XDREAD command (see 5.38 and 5.39) with starting LOGICAL BLOCK ADDRESS field and TRANSFER LENGTH field that match, or are a subset of, the LOGICAL BLOCK ADDRESS field and TRANSFER LENGTH field of this command.

5.41 XDWRITE (32) command

The XDWRITE (32) command (see table 80) requests that the device server:

1) read the specified logical block(s);
2) transfer logical blocks from the data-out buffer;
3) perform an XOR operation with the logical blocks transferred from the data-out buffer and the logical blocks read, storing the resulting XOR data in a buffer; and
4) if the DISABLE WRITE bit is set to zero, write the logical blocks transferred from the data-out buffer.

Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format. The resulting XOR data shall be retained as specified in 4.14.4.

If the RTO_EN bit is set to one in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE.

Table 80 — XDWRITE (32) command

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

See the XDWRITE (10) command (see 5.40) and SPC-3 for the definitions of the fields in this command.

5.42 XDWRITEREAD (10) command

The XDWRITEREAD (10) command (see table 81) requests that the device server:

1) read the specified logical block(s);
2) transfer logical blocks from the data-out buffer;
3) XOR the logical blocks transferred from the data-out buffer with the logical blocks read, storing the resulting XOR data in a buffer;
4) if the DISABLE WRITE bit is set to zero, write the logical blocks transferred from the data-out buffer; and
5) transfer the resulting XOR data to the data-in buffer.

Each logical block includes user data and may include protection information, based on the WRPROTECT field, the XORPINFO bit, and the medium format. This command is equivalent to an XDWRITE (10) command (see 5.40) followed by an XDREAD (10) command (see 5.38) specifying the same values for the GROUP NUMBER field, the LOGICAL BLOCK ADDRESS field and the TRANSFER LENGTH field. This command is only available on transport protocols supporting bidirectional commands.

If the RTO_EN bit is set to one in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE.

Table 81 — XDWRITEREAD (10) command

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

See the XDWRITE (10) command (see 5.40) and XDREAD (10) command (see 5.38) for the definitions of the fields in this command.

5.43 XDWRITEREAD (32) command

The XDWRITEREAD (32) command (see table 82) requests that the device server

1) read the specified logical block(s),
2) transfer logical blocks from the data-out buffer,
3) XOR the logical blocks transferred from the data-out buffer with the logical blocks read, storing the resulting XOR data in a buffer,
4) if the DISABLE WRITE bit is set to zero, write the logical blocks transferred from the data-out buffer and
5) transfer the resulting XOR data to the data-in buffer.

Each logical block includes user data and may include protection information, based on the WRPROTECT field, the XORPINFO bit and the medium format. This command is equivalent to an XDWRITE (32) command (see 5.41) followed by an XDREAD (32) command (see 5.39) specifying the same values for the GROUP NUMBER field, the LOGICAL BLOCK ADDRESS field and the TRANSFER LENGTH field. This command is only available on transport protocols supporting bidirectional commands.

If the RTO_EN bit is set to one in the READ CAPACITY (16) parameter data (see 5.11), 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 COMMAND OPERATION CODE.

Table 82 — XDWRITEREAD (32) command

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

See the XDWRITEREAD (10) command (see 5.42) and SPC-3 for the definitions of the fields in this command.

5.44 XPWRITE (10) command

The XPWRITE (10) command (see table 83) requests that the device server:

1) read the specified logical block(s);
2) transfer logical blocks from the data-out buffer; and
3) XOR the logical blocks transferred from the data-out buffer with the logical blocks read, writing the resulting XOR data.

Each logical block includes user data and may include protection information, based on the XORPINFO bit and the medium format.

Table 83 — XPWRITE (10) command

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

See the READ (10) command (see 5.6) for the definition of the DPO bit. See the WRITE (10) command (see 5.25) for the definitions of the FUA bit and the FUA_NV bit. See the PRE-FETCH (10) command (see 5.3) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.3) and 4.17 for the definition of the GROUP NUMBER field.

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

If the XORPINFO bit is set to one, the device server supports protection information, and the medium has been formatted with protection information, the device server shall XOR the user data and protection information transferred from the data-out buffer with the user data and protection information read and then write the resulting XOR data. The device server shall not check any of the protection information fields.

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

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

The TRANSFER LENGTH field specifies the number of contiguous logical blocks that shall be read, XORed with logical blocks transferred from the data-out buffer and written, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. If the logical block address plus the transfer length exceeds the capacity of the medium, 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 in the Block Limits VPD page (see 6.4.2).

5.45 XPWRITE (32) command

The XPWRITE (32) command (see table 84) requests that the device server:

1) read the specified logical block(s);
2) transfer logical blocks from the data-out buffer; and
3) XOR the logical blocks transferred from the data-out buffer with the logical blocks read, writing the resulting XOR data.

Each logical block includes user data and may include protection information, based on the XORPINFO bit and the medium format.

Table 84 — XPWRITE (32) command

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

See the XPWRITE (10) command (see 5.44) and SPC-3 for the definitions of the fields in this command.

6 Parameters for direct-access block devices

6.1 Diagnostic parameters

6.1.1 Diagnostic parameters overview

This subclause defines the descriptors and pages for diagnostic parameters used with direct-access block devices. The diagnostic page codes for direct-access block devices are defined in table 85.

Table 85 — Diagnostic page codes

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

6.1.2 Translate Address Output diagnostic page

The Translate Address diagnostic pages allow the application client to translate an address in one of the formats supported by the FORMAT UNIT command (see 5.2.2.4) (i.e., a short block format address, a long block format address, a physical sector format address, or a bytes from index format address) into any one of the other formats. The address to be translated is sent to the device server with the SEND DIAGNOSTIC command and the results are returned to the application client by the RECEIVE DIAGNOSTIC RESULTS command.

Table 86 defines the format of the Translate Address Output diagnostic page sent with the SEND DIAGNOSTIC command. The translated address is returned in the Translate Address Input diagnostic page (see 6.1.3).

Table 86 — Translate Address Output diagnostic page

Byte\Bit	7	6	5	4	3	2	1	0
0	PAGE CODE (40h)							
1	Reserved							
2	(MSB)							
3	PAGE LENGTH (000Ah)							(LSB)
4	Reserved					SUPPLIED FORMAT		
5	Reserved					TRANSLATE FORMAT		
6	(MSB)							
13	ADDRESS TO TRANSLATE							(LSB)

The SUPPLIED FORMAT field specifies the format of the ADDRESS TO TRANSLATE field. Valid values for this field are defined in the DEFECT LIST FORMAT field of the FORMAT UNIT command (see 5.2). If the device server does not support the requested format it 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 the device server shall use for the result of the address translation. Valid values for this field are defined in the DEFECT LIST FORMAT field of the FORMAT UNIT command. If the device server does not support the specified format it shall terminate the 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 which the application client is requesting the device server to translate. The format of this field depends on the value in the SUPPLIED FORMAT field. The formats are described in 5.2.2.4. If the short block format address descriptor is specified, 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 00000000h.

6.1.3 Translate Address Input diagnostic page

Table 87 defines the Translate Address Input diagnostic page retrieved with the RECEIVE DIAGNOSTIC RESULTS command after the Translate Address Output diagnostic page (see 6.1.2) has been sent with the SEND DIAGNOSTIC command. If a Translate Address Output diagnostic page has not yet been processed, the results of a RECEIVE DIAGNOSTIC RESULTS command requesting this diagnostic page are vendor-specific.

Table 87 — Translate Address Input diagnostic page

Byte\Bit	7	6	5	4	3	2	1	0
0	PAGE CODE (40h)							
1	Reserved							
2	(MSB)							
3	PAGE LENGTH (n - 3)							(LSB)
4	Reserved					SUPPLIED FORMAT		
5	RAREA	ALTSEC	ALTTRK	Reserved		TRANSLATED FORMAT		
	Translated address(es)							
6	(MSB)							
13	TRANSLATED ADDRESS 1							(LSB)
	...							
n - 7	(MSB)							
n	TRANSLATED ADDRESS x (if required)							(LSB)

The Translate Address Input diagnostic page contains a four-byte page header that specifies the page code and length followed by two bytes that describe the translated address followed by zero or more translated addresses.

The PAGE LENGTH field contains the number of parameter bytes that follow.

The SUPPLIED FORMAT field contains the value from the SUPPLIED FORMAT field in the previous Translate Address Output diagnostic page (see 6.1.2).

A reserved area (RAREA) bit set to zero indicates that no part of the translated address falls within a reserved area of the medium. A RAREA bit set to one indicates that all or part of the translated address falls within a reserved area of the medium (e.g., speed tolerance gap, alternate sector, or vendor reserved area). If the entire translated address falls within a reserved area, the device server may not return a translated address.

An alternate sector (ALTSEC) bit set to zero indicates that no part of the translated address is located in an alternate sector of the medium or that the device server is unable to determine this information. An ALTSEC bit

set to one indicates that the translated address is physically located in an alternate sector of the medium. If the device server is unable to determine if all or part of the translated address is located in an alternate sector it shall set this bit to zero.

An alternate track (ALTTRK) bit set to zero indicates that no part of the translated address is located on an alternate track of the medium. An ALTTRK bit set to one indicates that part or all of the translated address is located on an alternate track of the medium or the device server is unable to determine if all or part of the translated address is located on an alternate track.

The TRANSLATED FORMAT field contains the value from the TRANSLATE FORMAT field in the previous Translate Address Output diagnostic page (see 6.1.2).

The TRANSLATED ADDRESS field(s) contains the address(es) the device server translated from the address supplied by the application client in the previous Translate Address Output diagnostic page. Each field shall be in the format specified in the TRANSLATE FORMAT field. The formats are described in 5.2.2.4. If the short block format address descriptor is specified, the first four bytes of the TRANSLATED ADDRESS field shall contain the short block format address descriptor and the last four bytes shall contain 00000000h.

If the returned data is in short block format, long block format, or physical sector format and the ADDRESS TO TRANSLATE field in the previous Translate Address Output diagnostic page covers more than one address after it has been translated (e.g., because of multiple physical sectors within a single logical block or multiple logical blocks within a single physical sector) the device server shall return all possible addresses that are contained in the area specified by the address to be translated. If the returned data is in bytes from index format, the device server shall return a pair of translated values for each of the possible addresses that are contained in the area specified by the ADDRESS TO TRANSLATE field in the previous Translate Address Output diagnostic page. Of the pair of translated values returned, the first indicates the starting location and the second the ending location of the area.

6.2 Log parameters

6.2.1 Log parameters overview

This subclause defines the descriptors and pages for log parameters used with direct-access block devices. See SPC-3 for a detailed description of logging operations. The log page codes for direct-access block devices are defined in table 88.

Table 88 — Log page codes

Log page code	Description	Reference
00h	Supported Log Pages log page	SPC-3
01h	Buffer Over-Run/Under-Run log page	SPC-3
02h	Write Error Counter log page	SPC-3
03h	Read Error Counter log page	SPC-3
04h	Reserved	
05h	Verify Error Counter log page	SPC-3
06h	Non-Medium Error log page	SPC-3
07h	Last n Error Events log page	SPC-3
08h	Format Status log page	6.2.2
09h	Restricted (see SPC-3)	
0Ah	Restricted (see SPC-3)	
0Bh	Last n Deferred Errors Or Asynchronous Events log page	SPC-3
0Ch	Reserved	
0Dh	Temperature log page	SPC-3
0Eh	Start-Stop Cycle Counter log page	SPC-3
0Fh	Application Client log page	SPC-3
10h	Self-Test Results log page	SPC-3
11h - 16h	Reserved	
17h	Non-volatile Cache log page	6.2.3
18h	Protocol-Specific Port log page	SPC-3
19h - 2Eh	Reserved	
2Fh	Informational Exceptions log page	SPC-3
30h - 3Eh	Vendor-specific	
3Fh	Reserved	

6.2.2 Format Status log page

The Format Status log page (log page code 08h) captures the state of the direct-access block device since the most recent successful FORMAT UNIT command (see 5.2) was completed. Additionally, this log page provides Defect Management information for the device server.

The Format Status log page uses the log page format defined in SPC-3.

Table 89 defines the parameter codes for the Format Status log page.

Table 89 — Format Status log page parameter codes

Parameter code	Description
0000h	Format Data Out
0001h	Grown Defects During Certification
0002h	Total Blocks Reallocated During Format
0003h	Total New Blocks Reallocated
0004h	Power On Minutes Since Format
0005h - 7FFFh	Reserved
8000h - FFFFh	Vendor-specific

The PARAMETER LENGTH field of each log parameter (see SPC-3) contains the length of the corresponding PARAMETER VALUE field and is vendor-specific.

Event counts are returned as a result of the LOG SENSE command. The default value for each event count listed in table 89 shall be zero. Attempts to change these event counts by issuing a LOG SELECT with these fields set to non-zero values is not considered an error and shall have no effect on the saved values.

If information about a log parameter is not available, the device server shall return a value with each byte set to FFh (e.g., if the PARAMETER LENGTH field is set to 02h, the PARAMETER VALUE field is set to FFFFh). If the most recent FORMAT UNIT command failed, the device server shall return a value with each byte set to FFh for each log parameter.

The Format Data Out parameter contains the entire FORMAT UNIT parameter list (see 5.2.2) from the most recent successful FORMAT UNIT command. This includes:

a) the parameter list header;
b) the initialization pattern descriptor, if any; and
c) the defect list, if any.

The Grown Defects During Certification parameter is a count of the number of defects detected as a result of performing certification during processing of the most recent successful FORMAT UNIT command. This count reflects only those defects detected and replaced that were not already part of the PLIST or GLIST. If a certification pass was not performed the GROWN DEFECTS DURING CERTIFICATION field shall be set to zero.

The Total Blocks Reallocated During Format parameter is a count of the total number of logical blocks that were reallocated during the most recent successful FORMAT UNIT command.

The Total New Blocks Reallocated parameter is a count of the total number of logical blocks that have been reallocated since the completion of the most recent successful FORMAT UNIT command.

The Power On Minutes Since Format parameter represents 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 UNIT command.

Upon receiving the FORMAT UNIT command, the device server should set all parameters within the Format Status log page to indicate that no such information is available. Only upon successful completion of the FORMAT UNIT command should the device server update the affected fields.

The target save disable (TSD) bit in the PARAMETER CONTROL byte (see SPC-3) shall always be set to zero to indicate that the device server provides an implicit saving frequency.

NOTE Removable media device servers may save log page information with the medium in a vendor-specific manner and location.