

INTERNATIONAL STANDARD

ISO/IEC 14776-321

First edition
2002-08

Information technology – Small computer system interface-3 (SCSI-3) – Part 321: Block commands (SBC)



Reference number
ISO/IEC 14776-321:2002(E)

INTERNATIONAL STANDARD

ISO/IEC 14776-321

First edition
2002-08

Information technology – Small computer system interface-3 (SCSI-3) – Part 321: Block commands (SBC)

© ISO/IEC 2002

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

ISO/IEC Copyright Office • Case postale 56 • CH-1211 Genève 20 • Switzerland



PRICE CODE **XB**

For price, see current catalogue

CONTENTS

FOREWORD	8
INTRODUCTION	9
1 Scope	10
2 Normative references	12
2.1 Normative reference overview	12
2.2 Approved references	12
2.3 References under development	12
3 Definitions, symbols and abbreviations	13
3.1 Definitions	13
3.1.1 Definitions specific to direct access devices	13
3.1.2 Definitions specific to optical memory block devices and to write-once block devices	15
3.2 Symbols and abbreviations	15
3.3 Keywords	16
3.4 Conventions	17
4 General	17
5 SCSI block device models	18
5.1 Direct-access device type model	18
5.1.0 General	18
5.1.1 Removable medium	18
5.1.2 Logical blocks	19
5.1.3 Ready state	19
5.1.4 Power conditions	19
5.1.5 Initialization	21
5.1.6 Medium defects	22
5.1.7 Cache memory	22
5.1.8 Reservations	23
5.1.9 Seek(10)	25
5.1.10 Notched devices	25
5.1.11 Rotational position locking	25
5.1.12 Relative addressing	25
5.1.13 Error reporting	25
5.1.14 Examples	26
5.1.15 Model for XOR commands	27
5.2 Model for optical memory block devices	36
5.2.0 General	36
5.2.1 Defect management	37
5.2.2 Error reporting	37
5.3 Model for write-once block devices	38
5.3.0 General	38
5.3.1 Logical blocks	38
5.3.2 Initialization	39
5.3.3 Physical medium defects	39
5.3.4 Error reporting	39
6 Commands for block devices	40



6.1	Commands for direct-access block devices overview	40
6.1.1	Commands for direct-access block devices	40
6.1.2	FORMAT UNIT command	42
6.1.3	LOCK UNLOCK CACHE command	50
6.1.4	PRE-FETCH command	51
6.1.5	READ(6) command	52
6.1.6	READ(10) command	53
6.1.7	READ CAPACITY command	54
6.1.8	READ DEFECT DATA(10) command	55
6.1.9	READ LONG command	57
6.1.10	REASSIGN BLOCKS command	58
6.1.11	REBUILD command	60
6.1.12	REGENERATE command	62
6.1.13	SEEK(10) command	63
6.1.14	SET LIMITS(10) command	64
6.1.15	START STOP UNIT command	65
6.1.16	SYNCHRONIZE CACHE command	67
6.1.17	VERIFY command	68
6.1.18	WRITE(6) command	69
6.1.19	WRITE(10) command	70
6.1.20	WRITE AND VERIFY command	70
6.1.21	WRITE LONG command	71
6.1.22	WRITE SAME command	72
6.1.23	XDREAD command	73
6.1.24	XDWRITE command	74
6.1.25	XDWRITE EXTENDED command	75
6.1.26	XPWRITE command	76
6.2	Commands for optical memory block devices	77
6.2.0	General	77
6.2.1	ERASE(10) command	79
6.2.2	ERASE(12) command	80
6.2.3	MEDIUM SCAN command	81
6.2.4	READ(12) command	83
6.2.5	READ DEFECT DATA(12) command	84
6.2.6	READ GENERATION command	85
6.2.7	READ UPDATED BLOCK(10) command	86
6.2.8	SET LIMITS(12) command	87
6.2.9	UPDATE BLOCK command	88
6.2.10	VERIFY(10) command	89
6.2.11	VERIFY(12) command	90
6.2.12	WRITE(10) command	91
6.2.13	WRITE(12) command	91
6.2.14	WRITE AND VERIFY(10) command	92
6.2.15	WRITE AND VERIFY(12) command	93
6.3	Commands for write-once block devices	94
7	Parameters for block devices	96
7.1	Parameters for direct-access block devices	96
7.1.1	Diagnostic parameters	96
7.1.2	Log parameters	96

7.1.3	Mode parameters	103
7.1.4	Parameters for optical memory block devices	129
7.1.5	Parameters for write-once block devices	132
Annex A	(informative) XOR command examples	133
A.1	XOR annex overview	133
A.2	Storage array controller supervised XOR operations	133
A.2.1	Update write operation	133
A.2.2	Regenerate operation	134
A.2.3	Rebuild operation	135
A.3	Third-party XOR operations	136
A.3.1	Update write operation	136
A.3.2	Regenerate operation	137
A.3.3	Rebuild operation	138
A.4	Hybrid subsystem XOR operations	139
A.4.1	Regenerate operation	139
A.4.2	Rebuild operation	140
Bibliography		142
Figure 1	– SCSI standards – General structure	11
Figure 2	– SCSI power conditions flow control (automatic switching)	20
Figure 3	– SCSI power conditions flow control (controlled switching)	21
Figure 4	– Power conditions flowchart	119
Figure A.1	– Rebuild operation	134
Figure A.2	– Regenerate operation	135
Figure A.3	– Rebuild operation	136
Figure A.4	– Update write operation	137
Figure A.5	– Regenerate operation	138
Figure A.6	– Rebuild operation	139
Figure A.7	– Regenerate operation	140
Figure A.8	– Rebuild operation	141

Table 1 – Sample error conditions	26
Table 2 – Sample error conditions	38
Table 3 – Sample error conditions	39
Table 4 – Commands for direct-access block devices	40
Table 5 – FORMAT UNIT command.....	42
Table 6 – FORMAT UNIT parameter list.....	43
Table 7 – DEFECT LIST HEADER.....	44
Table 8 – FORMAT UNIT defect descriptor format and requirements	45
Table 9 – DEFECT DESCRIPTOR – Block format.....	47
Table 10 – DEFECT DESCRIPTOR – Bytes from index format.....	47
Table 11 – DEFECT DESCRIPTOR – Physical sector format	48
Table 12 – Initialization pattern descriptor.....	48
Table 13 – Initialization pattern modifier.....	49
Table 14 – Initialization pattern type.....	49
Table 15 – LOCK UNLOCK CACHE command	50
Table 16 – PRE-FETCH command.....	51
Table 17 – READ(6) command	52
Table 18 – READ(10) command	53
Table 19 – READ CAPACITY command.....	54
Table 20 – READ CAPACITY data	55
Table 21 – READ DEFECT DATA(10) command.....	55
Table 22 – READ DEFECT DATA(10) defect list	56
Table 23 – READ LONG command	57
Table 24 – REASSIGN BLOCKS command.....	58
Table 25 – REASSIGN BLOCKS defect list	59
Table 26 – REBUILD command	60
Table 27 – PORT CONTROL field.....	61
Table 28 – REBUILD and REGENERATE parameter data	61
Table 29 – SOURCE DESCRIPTOR format	62
Table 30 – REGENERATE command.....	63
Table 31 – SEEK(10) command.....	64
Table 32 – SET LIMITS(10) command	64
Table 33 – START STOP UNIT command.....	65
Table 34 – POWER CONDITIONS	66
Table 35 – SYNCHRONIZE CACHE command.....	67
Table 36 – VERIFY command.....	68
Table 37 – WRITE(6) command.....	69
Table 38 – WRITE(10) command.....	70
Table 39 – WRITE AND VERIFY command.....	71
Table 40 – WRITE LONG command	72
Table 41 – WRITE SAME command	73
Table 42 – XDREAD command.....	74
Table 43 – XDWRITE command	74

Table 44 – XDWRITE EXTENDED command	75
Table 45 – XPWRITE command	77
Table 46 – Commands for optical memory block devices	78
Table 47 – ERASE(10) command	80
Table 48 – ERASE(12) command	81
Table 49 – MEDIUM SCAN command	81
Table 50 – MEDIUM SCAN parameter list	82
Table 51 – READ(12) command	84
Table 52 – READ DEFECT DATA(12) command	84
Table 53 – READ DEFECT DATA(12) list header	85
Table 54 – READ GENERATION command	85
Table 55 – Maximum generation data block	86
Table 56 – READ UPDATED BLOCK(10) command	86
Table 57 – SET LIMITS(12) command	87
Table 58 – UPDATE BLOCK command	88
Table 59 – VERIFY command	89
Table 60 – VERIFY(12) command	90
Table 61 – WRITE(10) command	91
Table 62 – WRITE(12) command	92
Table 63 – WRITE AND VERIFY(10) command	92
Table 64 – WRITE AND VERIFY(12) command	94
Table 65 – Commands for write-once block devices	95
Table 66 – Diagnostic page codes	97
Table 67 – Translate address page – SEND DIAGNOSTIC	97
Table 68 – Translate address page – RECEIVE DIAGNOSTIC	98
Table 69 – Device status page – SEND DIAGNOSTIC	99
Table 70 – Device status page – RECEIVE DIAGNOSTIC	100
Table 71 – SYNCHRONIZATION field	100
Table 72 – Log page codes	101
Table 73 – Format status log page	102
Table 74 – Direct-access medium-type codes	103
Table 75 – Device specific parameter	103
Table 76 – Mode page codes	104
Table 77 – Caching page	105
Table 78 – Demand read retention priority and write retention priority	106
Table 79 – Flexible disk page	109
Table 80 – Examples of transfer rates	110
Table 81 – PIN 34 field	111
Table 82 – PIN 4 field	112
Table 83 – PIN 1 field	112
Table 84 – Format device page	113
Table 85 – Reporting of default sector formatting support	115
Table 86 – Reporting of changeable sector formatting support	115

Table 87 – Medium types supported page	115
Table 88 – Notch page	116
Table 89 – Power condition page	118
Table 90 – Read-write error recovery page	119
Table 91 – Error recovery bit definitions	121
Table 92 – Combined error recovery parameter descriptions	122
Table 93 – Rigid disk device geometry page	125
Table 94 – Rotational position locking	126
Table 95 – Verify error recovery page	127
Table 96 – XOR control mode page	128
Table 97 – Diagnostic page codes	129
Table 98 – Log page codes	129
Table 99 – Optical memory medium-type codes	130
Table 100 – Optical memory block device specific parameter	130
Table 101 – Optical memory density codes	131
Table 102 – Mode page codes	131
Table 103 – Optical memory page	132

.....

INFORMATION TECHNOLOGY – SMALL COMPUTER SYSTEM INTERFACE-3 (SCSI-3) –

Part 321: Block commands (SBC)

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 through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.
- 2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC1. 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) 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 14776-321 was prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

ISO/IEC 14776-321 is to be read in conjunction with ISO/IEC 14776-411, ISO/IEC 14776-311 and ISO/IEC 14776-351.¹⁾

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

¹⁾ For details, see clause 2.

INTRODUCTION

The SCSI command set described in this document is designed to provide efficient peer-to-peer operation of input/output logical units by an operating system using block transfers. The SCSI command set assumes an underlying command-response protocol.

This SCSI command set provides multiple operating systems concurrent control over one or more input/output logical units. However, the multiple operating systems are assumed to coordinate their actions properly to prevent data corruption. This SCSI standard provides commands that assist with coordination between multiple operating systems. However, details of the coordination are beyond the scope of the SCSI command set.

At the time this standard was developed SCSI included the following:

- physical interconnects;
- transport protocols;
- shared command set;
- architecture model;
- common access method.

Please refer to the bibliography for examples of international standards referring to the above items.

INFORMATION TECHNOLOGY – SMALL COMPUTER SYSTEM INTERFACE-3 (SCSI-3) –

Part 321: Block commands (SBC)

1 Scope

This part of ISO/IEC 14776 defines the command set extensions to facilitate operation of SCSI block devices.

It specifies the functional requirements for the SCSI-Block Command set (SBC). SBC permits that SCSI block logical units, such as flexible disks, rigid disks, optical disks, etc. be attached to computers, and it provides the definition for their use.

This standard defines a logical unit model for SCSI block logical units. Also defined are SCSI commands that apply to SCSI block logical units.

The clause(s) of this standard pertaining to the SCSI block device class, implemented in conjunction with the applicable clauses of ISO/IEC 14776-311, fully specify the standard command set for SCSI block devices.

The objectives of this standard are the following:

- a) permit an application client to communicate with a logical unit that declares itself to be a direct-access device, write-once device and optical memory device in the device type field of the INQUIRY command response data over an SCSI service delivery subsystem;
- b) define commands unique to the type of SCSI block devices;
- c) define commands to manage the operation of SCSI block devices;
- d) define the differences between types of SCSI block devices.

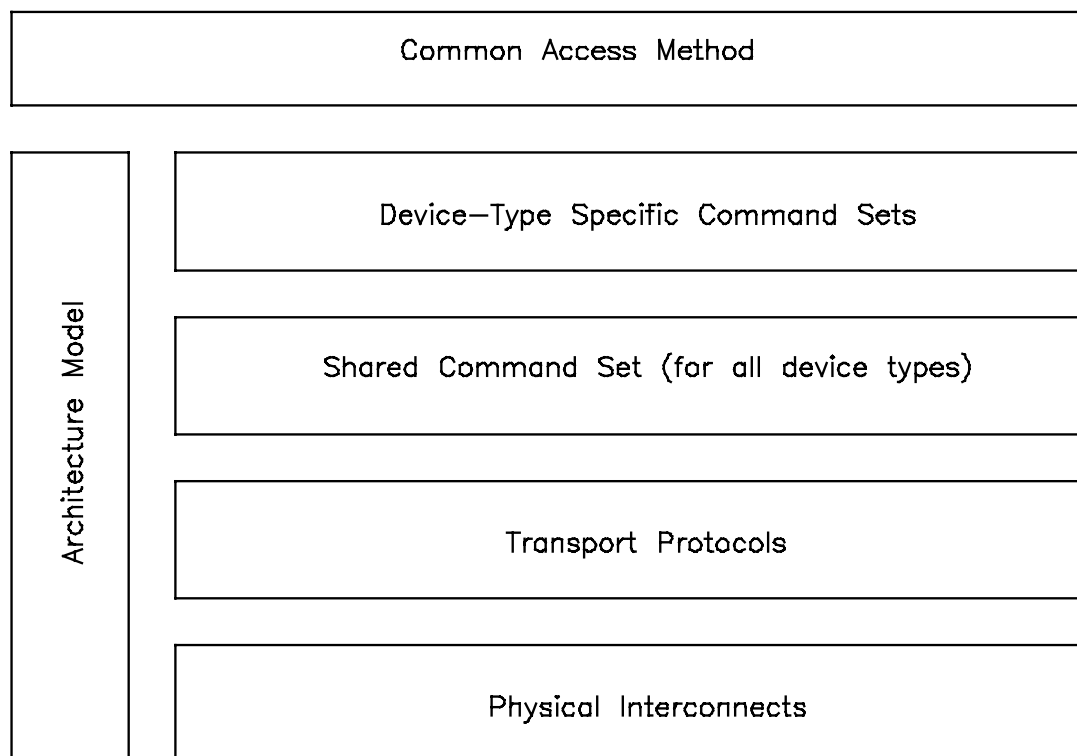


Figure 1 – SCSI standards – General structure

Figure 1 shows the general structure of SCSI standards. The figure does not imply a relationship such as a hierarchy, protocol stack, or system architecture. It indicates the applicability of a standard to the implementation of a given transport.

2 Normative references

2.1 Normative reference overview

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.

NOTE The term SCSI is used wherever it is not necessary to distinguish between the different types of SCSI.

2.2 Approved references

ISO/IEC 9316:1995, *Information technology – Small Computer System Interface-2*

NOTE ISO/IEC 9316 is referred to herein as SCSI-2. The term SCSI-3 in this standard refers to different versions of SCSI defined since SCSI-2.

ISO/IEC 13614:1995, *Information technology – Interchange on 300 mm optical disk cartridges of the write once read multiple (WORM) type using the SSF method*

ISO/IEC 14776-341:2000, *Information technology – Small Computer System Interface-3 (SCSI-3) – Part 341: Controller Commands (SCC)*

ISO/IEC 14776-411:1999, *Information technology – Small Computer System Interface (SCSI-3) – Part 411: SCSI-3 Architecture Model (SCSI-3 SAM)*

ISO/IEC 10090:1992, *Information technology – 90 mm optical disk cartridges, rewritable and read only, for data interchange*

2.3 References under development

At the time of publication, the following referenced standards were still under development.

ISO/IEC 14776-311, – *Information technology – Small Computer System Interface-3 (SCSI-3) – Part 311: Primary Commands*

ISO/IEC 14776-351, – *Information technology – Small Computer System Interface-3 (SCSI-3) – Part 351: Medium Changer Commands*

ISO/IEC 14776-362, – *Information technology – Small Computer System Interface (SCSI 3) – Part 362: Multimedia commands-2 (MMC-2)*

3 Definitions, symbols and abbreviations

3.1 Definitions

3.1.1 Definitions specific to direct access devices

3.1.1.1

block device

a device that is capable of containing data stored in blocks that have a unique logical block address

3.1.1.2

cache memory

a temporary (and often volatile) data storage area outside the user-accessible area that may contain a subset of the data stored in the non-volatile data storage area

A cache memory is usually faster to access than the medium and thus has the effect of increasing data throughput by reducing the number of accesses to the medium.

3.1.1.3

check data

information contained within a redundancy group that allows lost or destroyed user data to be recreated

3.1.1.4

data-in buffer

buffer identified by the application client to receive data from the device server during the execution of a command

3.1.1.5

data-out buffer

buffer identified by the application client to supply data that is sent from the application client to the device server during the execution of a command

3.1.1.6

domain

an I/O system consisting of a set of SCSI devices that interact with one another by means of a service delivery subsystem

3.1.1.7

exclusive-or

logical function that combines two logical inputs producing a logical output true state if one but not both inputs are true

This function is used in error correction algorithms. In this standard the term encompasses the entire algorithm but does not define the specific polynomial. The exclusive-or operation may be performed by the storage array controller or by the storage device.

3.1.1.8

extent

an extent is a specified number of logical blocks and all or part of a block device

3.1.1.9

host

any combination of initiators and application clients that form a device managing one or more peripheral devices

3.1.1.10

logical block

unit of data supplied or requested by an initiator

3.1.1.11

non-volatile medium

physical storage medium that retains data written to it for a subsequent read operation through power off/on cycles

An example of a non-volatile medium is a disk within a device that stores data as magnetic field changes which do not require device power to exist.

3.1.1.12

notch

all or part of the medium having a consistent set of geometry parameters

Notches are used to increase storage capacity by optimizing the number of bytes per track between the inner and outer tracks.

3.1.1.13

redundancy group

grouping of protected space and associated check data into a single type of data redundancy

This standard only supports the exclusive-or type of redundancy.

[see ISO/IEC 14776-341:2000]

3.1.1.14

storage array controller

any combination of an initiator and application clients that originates SCSI command descriptor blocks and performs the services of a SACL

A storage array controller organizes a group of storage devices into various objects (for example, redundancy groups, volume sets, etc.).

[see ISO/IEC 14776-411:1999]

3.1.1.15

storage array conversion layer (SACL)

converts input logical unit numbers to output logical unit numbers and may convert input logical block addresses to output logical block addresses

3.1.1.16

third party

when used in conjunction with exclusive-or operations refers to the operations performed by a primary target with a secondary target on behalf of the host storage array controller

3.1.1.17

user-accessible

the area of the medium that can be read from or written to by READ and WRITE commands

3.1.1.18

user data

the addressable logical blocks that are input to the SACL

Check data is not part of the addressable logical blocks.

3.1.1.19**volatile medium**

medium that does not retain data written to it for a subsequent read operation through power off/on cycles

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

3.1.2 Definitions specific to optical memory block devices and to write-once block devices**3.1.2.1****blank**

the logical block contains no information detectable by the block device, or is written with a pattern that appears to the block device as no data present. The logical block is considered ready for a write operation

3.1.2.2**generation**

indicates a relative revision level of a logical block that has been updated via the UPDATE BLOCK command

A logical block that has never been updated has only one generation associated with it.

3.1.2.3**read-only medium**

medium which is not to be written by the application client

The medium contains data prepared in a manner not defined by this standard.

3.1.2.4**update**

to write new data to a logical block without destroying the previous data

After a block has been updated, a normal read returns the most recent generation of the data. Earlier generations are still available after the update.

3.1.2.5**write-once medium**

medium which is to be written only once by any application client

Logical blocks on write-once media that have not been written are considered blank. Logical blocks on write-once media that have been written are not to be written again.

3.2 Symbols and abbreviations

CDB	command descriptor block
I/O	input/output
ID	identifier
LSB	least significant bit
MMC-2	SCSI-3 Multimedia Command Set-2 [see ISO/IEC 14776-362]
MSB	most significant bit
SAM	SCSI-3 Architecture Model [see ISO/IEC 14776-411]
SCC	SCSI-3 Controller Commands [see ISO/IEC 14776-341]
SCSI	either SCSI-2 or SCSI-3
SCSI-2	the Small Computer System Interface-2 [see ISO/IEC 9316:1996]
SCSI-3	the Small Computer System Interface-3
SPC	SCSI-3 Primary Command Set standard [see ISO/IEC 14776-311]
XOR	exclusive-or

3.3 Keywords

Several keywords are used to differentiate between different levels of requirements and optionality, as follows:

3.3.1

expected

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

3.3.2

mandatory

indicates items required to be implemented as defined by this standard

3.3.3

may

indicates flexibility of choice with no implied preference

3.3.4

obsolete

indicates items that were defined in prior SCSI standards but have been removed from this standard

3.3.5

optional

describes features that are not required to be implemented by this standard

However, if any optional feature defined by the standard is implemented, it shall be implemented as defined by this standard.

3.3.6

reserved

refers to bits, bytes, words, fields and code values that are set aside for future standardization

Their use and interpretation may be specified by future extensions to this or other standards. A reserved bit, byte, word or field shall be set to zero, or in accordance with a future extension to this standard. The recipient may not check reserved bits, bytes, words or fields. Receipt of reserved code values in defined fields shall be treated as an error.

3.3.7

shall

indicates a mandatory requirement

Designers are required to implement all such mandatory requirements to ensure interoperability with other standard conformant products.

3.3.8

should

indicates flexibility of choice with a strongly preferred alternative

Equivalent to the phrase "it is recommended".

3.3.9

vendor-specific

items (e.g., a bit, field, code value, etc.) that are not defined by this standard and may be vendor defined.

3.4 Conventions

Lower case is used for words having the normal English meaning. Certain words and terms used in this standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in clause 3 or in the text where they first appear.

Listed items in this standard do not represent any priority. Any priority is explicitly indicated. Formal lists (for example, (a) red; (b) blue; (c) green) connoted by letters are in an arbitrary order. Formal lists (for example, (1) red; (2) blue; (3) green) connoted by numbers are in a required sequential order.

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

The ISO/IEC convention of numbering is used (for example, the thousands and higher multiples are separated by a space, and a decimal comma is used instead of the decimal point, as shown in the following examples: 65 536 or 0,5).

The additional conventions are:

- a) the names of abbreviations, commands and acronyms used as signal names are in uppercase (for example, IDENTIFY DEVICE);
- b) fields containing only one bit are referred to as the "NAME" bit instead of the "NAME" field;
- c) field names are in SMALL CAPS to distinguish them from normal English;
- d) numbers that are not immediately followed by lower-case b or h are decimal values;
- e) numbers immediately followed by lower-case b (xxb) are binary values;
- f) numbers immediately followed by lower-case h (xxh) are hexadecimal values;
- g) the most significant bit of a binary quantity is shown on the left side and represents the highest algebraic value position in the quantity;
- h) if a field is specified as not meaningful or to be ignored, the entity that receives the field shall not check that field.

4 General

SCSI devices that conform to this standard are referred to as SCSI block devices. This includes the category of logical units commonly referred to as flexible disks, rigid disks, removable rigid disks, erasable optical discs, write once optical discs and read only optical discs. The ISO/IEC 14776-362, SCSI-3 Multimedia Command Set - 2 standard is typically used by CD-ROM devices.

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

Each logical block has a block length associated with it. This means that the block length for the medium can change from logical block to logical block. However, for simplicity the block length typically remains constant over the entire capacity of the medium.

5 SCSI block device models

5.1 Direct-access device type model

5.1.0 General

Direct-access block devices store blocks of data for later retrieval. Each block of data is stored at a unique logical block address. An application client issues WRITE commands to store the blocks of data (write operations) and READ commands to retrieve the blocks of data (read operations). Other commands issued by the application client may also cause write and read operations to occur. A write operation causes one or more blocks of data to be written on the medium. A read operation causes one or more blocks of data to be read from the medium. A verify operation confirms that one or more blocks of data were correctly written and can be read without error from the medium.

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

5.1.1 Removable medium

5.1.1.0 General

The medium may be removable (for example, used in a floppy disk device) or non-removable (for example, used in a fixed disk device). The removable medium may be contained within a cartridge (or jacket) to prevent damage to the recording surfaces. The combination of medium and cartridge is often called a removable volume.

A removable volume has an attribute of being mounted or de-mounted on a suitable transport mechanism. A removable volume is mounted when the direct access block device is capable of performing write or read operations to the medium. A mounted removable volume may not be accessible by an initiator if it is reserved by another initiator. A removable volume is de-mounted at any other time (for example, during loading, unloading or storage).

An application client may check whether a removable volume is mounted by issuing a TEST UNIT READY command. A volume that is loaded may need a START STOP UNIT command issued to become accessible for write or read operations.

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

5.1.1.1 Removable medium with an attached medium changer

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

The block device indicates its ability to support these commands by setting the MCHNGR bit to one in its standard INQUIRY data. An MCHNGR bit of one indicates that the MOVE MEDIA and READ ELEMENT STATUS commands are supported. Only one medium transport element is permitted (element 0) and only one data transfer element is permitted.

5.1.2 Logical blocks

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

The address of the first logical block is zero. The address of the last logical block is $[n - 1]$, where $[n]$ is the number of logical blocks available to the application client on the medium. A READ CAPACITY command may be issued to determine the value of $[n - 1]$. If a command is issued that requests access to a logical block not within the capacity of the medium, the command is terminated with CHECK CONDITION status and the additional sense key shall be set to LOGICAL BLOCK ADDRESS OUT OF RANGE with the additional sense code set to ILLEGAL REQUEST.

The number of bytes of data contained in a logical block is the block length. Each logical block has a block length associated with it. The block descriptor in the MODE SENSE data describes the block lengths that are used on the medium. The FORMAT UNIT command may be required to change the block length of block devices that support variable block lengths.

The location of a logical block on the medium is not required to have a relationship to the location of any other logical block. However, in a typical block device the logical blocks are located in an ascending order. 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 issued with a PMI bit of one may be useful in determining where longer access times occur.

5.1.3 Ready state

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

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

5.1.4 Power conditions

The optional power conditions permit the application client to modify the behavior of a block device in a manner that may reduce the required power. There is no notification to the application client that a block device has entered one of the power conditions. The power conditions may be controlled by the START STOP UNIT command or the power condition page of the MODE SELECT command. If both methods are being used on the same target/logical unit combination then any START STOP UNIT commands power condition request shall override the power condition page's power control. See the START STOP UNIT command description and the MODE SELECT power condition page description for more information. (See 6.1.15 and 7.1.3.6.)

No power condition shall affect the SCSI bus.

The lowest power consumption, with power applied, occurs in the Sleep condition. When in the Sleep condition a block device requires a WAKEUP task management function to be activated.

In the Standby condition a block device is capable of accepting commands, but media is not immediately accessible (for example, the spindle is stopped).

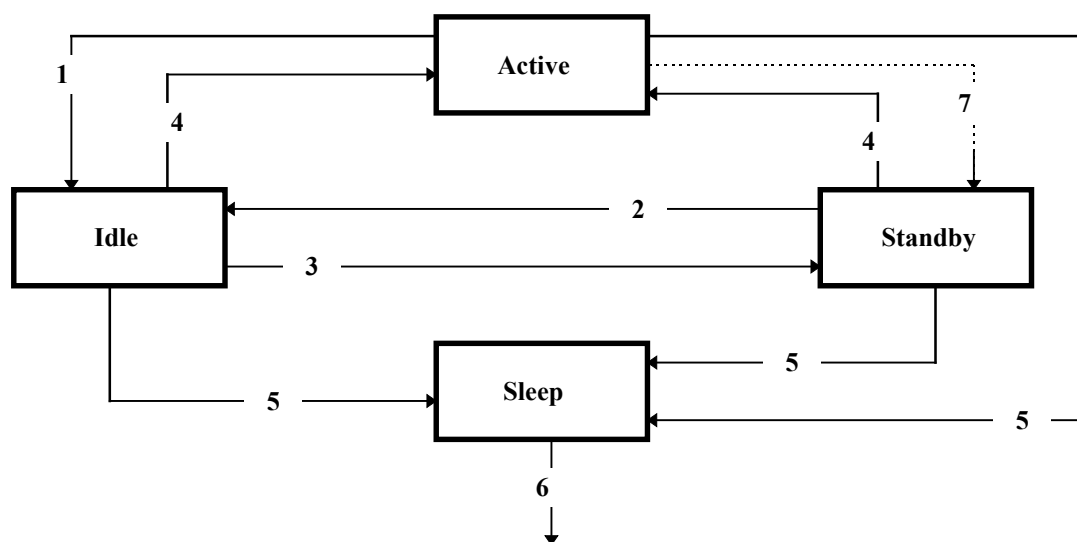
In the Idle condition, a block device is capable of responding quickly to media access requests. However, a block device in the idle condition may take longer than in the active condition, to complete the execution of a command because it may have to activate some circuitry.

In the Active condition a block device is capable of responding immediately to media access requests, and operations complete execution in the shortest time compared to the other power conditions.

Block devices that contain cache memory shall implicitly perform a SYNCHRONIZE CACHE command for the entire medium prior to entering any power condition that prevents access the media (for example, the spindle being stopped).

If implemented, the block device shall use the optional power condition page to control the power conditions after a power on or a WAKEUP task management function until a START STOP UNIT command is received with the POWER CONDITIONS field set to a value other than 0h or 7h. See 6.1.15 and 7.1.3.6.

Figure 2 shows the flow control between the different power conditions in a device that is setup to adjust itself automatically to the power condition that allows any command to execute.

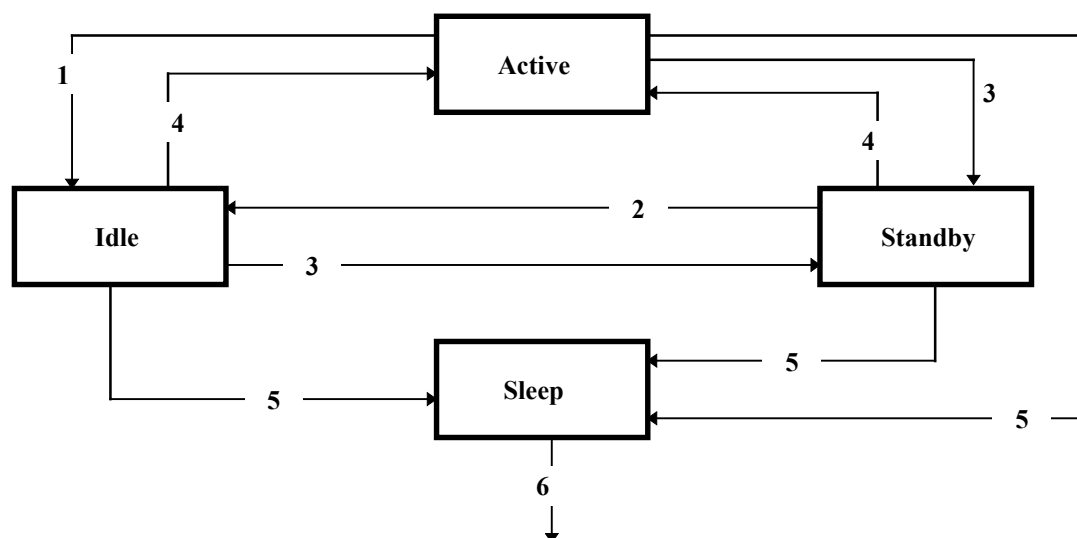


NOTE For Active, Idle or Standby, see f).

- a) Path 1: An idle time-out or a START STOP UNIT command with a power condition code of Ah.
- b) Path 2: Any command that can be executed within the power constraints of the Idle power condition.
- c) Path 3: A standby time-out or a START STOP UNIT command with a power condition code of Bh.
- d) Path 4: Any command that exceeds the power constraints of the Idle power condition.
- e) Path 5: A START STOP UNIT command with a power condition code 5h.
- f) Path 6: A WAKEUP task management function returns the device to the state defined by the saved power mode page parameters.
- g) Path 7: This path only occurs if the idle bit is set equal to zero.

Figure 2 – SCSI power conditions flow control (automatic switching)

Figure 3 shows the flow control between the different power conditions in a device that is setup to only allow changing of the power condition by the application client. Any command received that requires more power than allowed by the most recent power condition setting shall be terminated with a sense key of ILLEGAL REQUEST and the additional sense code shall be set to LOW POWER CONDITION ACTIVE.



NOTE Active, Idle or Standby - see f).

- a) Path 1: A START STOP UNIT command with a power condition code of 2h.
- b) Path 2: A START STOP UNIT command with a power condition code of 2h.
- c) Path 3: A START STOP UNIT command with a power condition code of 3h.
- d) Path 4: A START STOP UNIT command with a power condition code of 1h.
- e) Path 5: A START STOP UNIT command with a power condition code of 5h.
- f) Path 6: A WAKEUP task management function returns the device to the state defined by the saved power mode parameters.

Figure 3 – SCSI power conditions flow control (controlled switching)

5.1.5 Initialization

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

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 power-on and/or reset. A catastrophic failure of the direct-access block device may require the FORMAT UNIT command to be reissued.

Block devices that use a volatile medium may need to be initialized at each power-on and/or reset prior to the execution of read or write operations. Mode parameters may also need initialization.

5.1.6 Medium defects

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

Defects may also be detected and managed during execution of the FORMAT UNIT command. The FORMAT UNIT command defines four sources of defect information. These defects may be reassigned or avoided during the initialization process so that they do not appear in a logical block.

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

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

5.1.7 Cache memory

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

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

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

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

Sometimes the application client may want to have the blocks of data read from the medium instead of from the cache memory. The force unit access (FUA) bit is used to indicate that the device server shall access the physical medium. For a write operation, setting FUA to one causes the device server to complete the data write to the physical medium before completing the command. For a read operation, setting FUA to one causes the logical blocks to be retrieved from the physical medium.

When the DPO and FUA bits are both one, write and read operations, in effect, bypass the cache memory.

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

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

- a) the LOCK UNLOCK CACHE command controls whether certain logical blocks shall be held in the data cache for future use. Locking a logical block prevents its replacement by a future access. Unlocking a logical block exposes it to possible replacement by a future access (see 6.1.3);
- b) the PRE-FETCH command causes a set of logical blocks requested by the application client to be read into the data cache for possible future access. The blocks fetched are subject to later replacement unless they are locked (see 6.1.4);
- c) the SYNCHRONIZE CACHE command forces any pending write data in the requested set of logical blocks to be stored in the physical medium. This command may be used to ensure that the data was written and any detected errors reported (see 6.1.16);
- d) the MODE SELECT command defines a page for the control of cache behavior and handles certain basic elements of cache replacement algorithms (see ISO/IEC 14776-311).

5.1.8 Reservations

The access enabled or access disabled condition determines when an application client may store or retrieve user data on all or part of the medium. Access may be restricted for read operations, write operations, or both. This attribute may be controlled by an external mechanism or by the RESERVE and RELEASE commands (see ISO/IEC 14776-311).

The RESERVE and RELEASE commands define how different types of restricted access may be achieved, and to whom the access is restricted. This subclause describes the interaction of the application client that requested the reservation, as well as the other application clients.

Reservations are further controlled by the optional PERSISTENT RESERVE IN and PERSISTENT RESERVE OUT commands. For the requirements of this standard, reservations and releases made by use of the PERSISTENT RESERVE IN and PERSISTENT RESERVE OUT commands are the same as those using the RESERVE and RELEASE commands. See ISO/IEC 14776-311 for a description and the requirements of the various reservation commands.

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

The following paragraphs explain, on a command by command basis, the appropriate device server response when a reservation exists. Unless otherwise noted, the appropriate response to an application client that issues a command to a device server that is reserved to another initiator is RESERVATION CONFLICT status.

If any initiator has an extent reservation on a logical unit, no other initiator may affect the operating definition of that initiator by use of the CHANGE DEFINITION command. If the device server allows different operating definitions for each initiator, then there is no conflict. Otherwise, a reservation conflict occurs.

The COMPARE, COPY and COPY AND VERIFY commands are evaluated for reservation conflict as if they were normal write and read operations even when a device server is requested to copy to or from itself. For example, if a COPY is issued to logical unit 0 that requests the device server to copy from logical unit 0 to logical unit 1, access to logical unit 1 is also to be evaluated for reservation conflicts.

The FORMAT UNIT, PREVENT ALLOW MEDIUM REMOVAL (with a prevent bit of one), REZERO UNIT, and START STOP UNIT commands return a RESERVATION CONFLICT status if any other initiator has an extent reservation on the device server.

The INQUIRY and REQUEST SENSE commands are not affected by any kind of reservation.

The LOG SELECT, LOG SENSE, MODE SENSE, TEST UNIT READY, READ CAPACITY (PMI equal zero), READ BUFFER, WRITE BUFFER and READ DEFECT DATA commands are not affected by extent reservations.

The SEEK, LOCK UNLOCK CACHE, PRE-FETCH and SYNCHRONIZE CACHE commands are evaluated for reservation conflict as if they were normal write or read operations.

If an initiator has an extent reservation on a device server, and another initiator attempts a MODE SELECT command, a reservation conflict occurs if the command affects the manner of accessing the extent by the first initiator. If the command does not affect access to the extent, or parameters are saved for each initiator, then a reservation conflict does not occur.

The SEND DIAGNOSTIC and RECEIVE DIAGNOSTIC RESULTS commands conflict with an extent reservation only if they affect access to the extent.

The REASSIGN BLOCKS command shall not reassign a block that is in an extent reserved to another initiator.

The SET LIMITS command generates a reservation conflict if the logical blocks specified are within an extent reserved to another initiator.

All other commands that request read or write operations are evaluated for reservation conflict as described in the reserve commands.

When a system is integrated with more than one application client, agreement is required between the application clients as to how media is reserved and released during operations. Otherwise, an application client may be locked out of access to a logical unit in the middle of an operation. For example, an application client in initiator 'A' has a write operation in progress to a direct-access logical unit that has data stored in cache memory. Then, another application client in initiator 'B' issues a RESERVE command to the direct-access block device. As a result, the application client in initiator 'A' is locked out of issuing a SYNCHRONIZE CACHE command to ensure the integrity of the data. To prevent this from happening, an application client may issue a RESERVE command prior to the WRITE command.

5.1.9 Seek(10)

The SEEK(10) command provides a way for the application client to position the actuator of the block device in preparation for access to a particular logical block at some later time. Since this positioning action is implicit in other commands, the SEEK(10) command may not be useful with some direct-access block devices.

5.1.10 Notched devices

A notched (also known as zoned) device has areas of the medium with geometry changes. In the simplest case, the entire medium consists of a single notch. Multiple notches are often used to increase capacity of the device. On a disk, the inner tracks are physically shorter than the outer tracks. As a result, if each track is made to store the same number of data bits, the data is packed more densely on the inner tracks than the outer tracks. By using notches, the outer tracks may contain a different number of sectors than the inner tracks, while balancing the data density. This results in increased capacity.

The notch page is used to indicate the notch for assignment of values to the parameters in the format device page. By sequencing the notch page through each notch, the format device parameters of each notch are set. This may be done prior to initialization by the FORMAT UNIT command.

5.1.11 Rotational position locking

Rotational position locking is an optional feature implemented in some direct-access block devices to allow the synchronization of spindles between a number of logical units. The rotational position offset feature allows block devices to synchronize spindles at offsets from index. This may be useful in improving performance in systems that implement arrays of logical units.

5.1.12 Relative addressing

Relative addressing is a technique that may be useful in accessing structured data in a uniform manner. Relative addressing may be used when commands are linked.

The SET LIMITS command (see 6.1.14) is provided to define the limits of a linked chain of relative addressing commands. This protects against exceeding the specified set of blocks. The SET LIMITS command has no effect on any other initiator.

5.1.13 Error reporting

If any of the following conditions occur during the execution of a command (see Table 1), the command shall be terminated with CHECK CONDITION status and the sense key shall be set to the appropriate sense key with the appropriate additional sense code for the condition. Some errors may occur after the completion status has already been reported. For such errors, ISO/IEC 14776-311 defines a deferred error reporting mechanism. The following list illustrates some error conditions and the applicable sense keys. The list does not provide an exhaustive enumeration of all conditions that may cause the CHECK CONDITION status.

Table 1 – Sample error conditions

Condition	Sense key
Invalid logical block address	ILLEGAL REQUEST
Unsupported option requested	ILLEGAL REQUEST
Logical unit reset 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
Overrun or other error that might be resolved by repeating the command	ABORTED COMMAND
Attempt to write on write protected medium	DATA PROTECT

In the case of an invalid logical block address, the SENSE DATA INFORMATION field shall be set to the logical block address of the first invalid address.

In the case of an attempt to read a blank or previously unwritten block, the INFORMATION field shall be set to the logical block address of the first blank block encountered. The data read up to that block shall be transferred (optical memory and write-once block devices only).

In the case of an attempt to write a previously written block when blank checking is enabled, the INFORMATION field shall be set to the logical block address of the first non-blank block encountered (optical memory and write-once block devices only).

5.1.14 Examples

5.1.14.0 General

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

5.1.14.1 Rotating media

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

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

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

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

A disk device may have to be formatted prior to the initial access. Exceptions to this are devices that are formatted at the factory and some optical devices with pre-formatted media. A disk device format may create headers for each sector and initialize the data field. The MODE SELECT command is often used prior to formatting to establish the geometry (number of heads and tracks, sectors per track, etc.) and defect management scheme. Disk devices are usually non-volatile.

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

5.1.14.2 Sequential media

Some tape logical units are implemented as a direct access block device so that they may be used in disk oriented operating system environments. These logical units are sometimes referred to as random access tape or floppy tape. These logical units might be thought of as a disk device with one or more long tracks. Access time to a logical block is usually longer than for a disk device, since the tape requires that it be fast forwarded or rewound to the block. As a result, the SEEK command often is more useful for a tape than for a disk. The only way an application client may determine if a direct-access block device is a tape is by using the medium type code returned by the MODE SENSE command.

5.1.14.3 Memory media

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

These logical units may be non-mechanical and therefore the entire physical medium may be accessed in virtually the same access time. The data may be accessed as a bit or byte and this also speeds access time. Memory block devices may store less data than disks or tapes and are usually volatile (except Flash) when not protected by battery backup.

5.1.15 Model for XOR commands

5.1.15.0 General

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

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

- a read transfer from the device containing protected data;
- a write transfer to the device containing protected data;
- a read transfer from the device containing check data;
- a write transfer to the device containing check data.

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

In contrast, during storage array controller supervised XOR operations (see 5.1.15.1) only three data transfer operations are needed:

- a write transfer to the device containing protected data;
- a read transfer from the device containing protected data;
- a write transfer to the device containing check data.

During third party XOR operations (see 5.1.15.2) only two data transfer operations are needed:

- a write transfer from the storage array controller to the device containing protected data, and
- a write transfer from the device containing protected data to the device containing check data.

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

5.1.15.1 Storage array controller supervised XOR operations

5.1.15.1.0 General

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

5.1.15.1.1 Update write operation

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

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

5.1.15.1.2 Regenerate operation

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

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

5.1.15.1.3 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 sequence is as follows:

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

5.1.15.2 Third party XOR operations

5.1.15.2.0 General

Five XOR commands are needed to implement the third party XOR operations: XDWRITE EXTENDED, XPWRITE, XDREAD, REGENERATE and REBUILD. The storage array controller also uses READ and WRITE commands for certain operations. Third party XOR operations are restricted to systems where all devices involved in the operations are located in the same SCSI domain since direct interaction between those devices is required.

5.1.15.2.1 Update write operation

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

- a) an XDWRITE EXTENDED command is sent to the device containing protected data. This transfers the new write data to that device. The device reads its old user data, performs an XOR operation using the old user data and the received user data, temporarily stores the XOR result and writes the received user data to the medium;
- b) the device containing protected data becomes a temporary initiator and sends an XPWRITE command to the device containing check data. This transfers the resulting XOR data from the device containing protected data to the device containing check data. The device containing check data reads its check data, performs an XOR operation using the check data and the received XOR data and writes the resulting XOR result to the medium;
- c) after the device containing protected data receives status for its XPWRITE command, it returns ending status for the XDWRITE EXTENDED command to the storage array controller. This indicates that the operations on both the device containing protected data and the device containing check data have completed.

5.1.15.2.2 Regenerate operation

The regenerate operation is used to recreate a data block that is not readable from a data device. This is done by reading the associated data block from each of the other devices within the redundancy group and performing an XOR operation with each of these data blocks. The last XOR result is the regenerated data that had the error. The number of steps is dependent on the number of devices in the redundancy group, but the sequence is as follows (since XOR operands are commutable the XOR order is irrelevant and the order of steps 2 and 3 is interchangeable):

- a) a REGENERATE command is sent to a valid device in the redundancy group (a valid device is any device in the group other than the failed device). This transfers the regenerate parameter list from the storage array controller to the device;
- b) the device reads the requested data from its own medium. The device retains the requested data for a subsequent XOR operation;
- c) the device becomes a temporary initiator and sends a READ command to another device included in the regenerate parameter list. That target device transfers the requested data to the temporary initiator. The temporary initiator performs an XOR operation between the read data from the previous step and the read data received in this step. The resulting XOR data is retained for the next step;
- d) step 3 is repeated until all devices listed in the regenerate parameter list have been accessed. The last XOR data result is retained by the temporary initiator. The temporary initiator returns completion status for the REGENERATE command to the storage array controller when this regenerated user data is available;
- e) an XDREAD command is sent from the storage array controller to the device that had been a temporary initiator in the above steps. This transfers the regenerated user data from the last XOR data result from the device to the storage array controller.

5.1.15.2.3 Rebuild operation

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

- a) a REBUILD command is sent to the replacement device in the redundancy group (the device that replaces a failed device). This transfers the rebuild parameter list from the storage array controller to the device;

- b) the device becomes a temporary initiator and sends a READ command to a device included in the rebuild parameter list. That target device transfers the requested data to the temporary initiator. The temporary initiator retains this data for a subsequent XOR operation;
- c) the temporary initiator sends a READ command to another device included in the rebuild parameter list. That device transfers the requested data to the temporary initiator. The temporary initiator performs an XOR operation between the read data from the previous step and the read data received in this step. The resulting XOR data is retained for the next step;
- d) step 3 is repeated until all devices listed in the rebuild parameter list have been accessed. The last XOR data result is written to the replacement device's medium, then the device returns completion status for the REBUILD command to the storage array controller.

5.1.15.3 Hybrid subsystem XOR operations

5.1.15.3.0 General

In a hybrid subsystem the redundancy group is divided between two or more domains (see 5.1.15.1) and at least one of those domains contains two or more of the devices in the redundancy group. Such a system could do its XOR operations as described in 5.1.15.1 (Storage array controller supervised XOR operations), but it may choose to use third party XOR commands for parts of the XOR operation where all the involved devices are in the same domain. This subclause describes use of the third party XOR operations on a hybrid system with two domains. For illustration the redundancy group has six devices, with three devices in each domain.

5.1.15.3.1 Update write operation

When the update write operation involves two devices that are in different domains, the storage array controller uses the technique described in storage array controller supervised XOR operations. When the update write operation involves two devices that are in the same domain the storage array controller may use either the storage array controller supervised operation or the third party XOR operation.

5.1.15.3.2 Regenerate operation

The regenerate operation, for the illustrated case, always involves five XOR devices (all but the failed device) where three devices are in one domain (referred to as domain A) and two devices are in the other domain (referred to as domain B). The sequence is as follows (since XOR operands are commutable the XOR order is irrelevant and the order of steps 2 and 3 is interchangeable. Likewise, the order of steps 7 and 8 is interchangeable):

- a) a REGENERATE command is sent to a device in domain A. This transfers the regenerate parameter list (containing the other two valid devices and data extents) from the storage array controller to the device;
- b) the device reads the requested data from its own medium. The device retains the requested data for a future XOR operation;
- c) the device becomes a temporary initiator and sends a READ command to another device included in the regenerate parameter list. That device transfers the requested data to the temporary initiator. The temporary initiator performs an XOR operation between the read data from the previous step and the read data received in this step. The resulting XOR data is retained for the next step;
- d) step 3 is repeated until all devices listed in the regenerate parameter list have been accessed. The last XOR data result is retained by the temporary initiator. The temporary initiator returns completion status for the REGENERATE command to the storage array controller when this partially regenerated user data is available;

- e) an XDREAD command is sent from the storage array controller to the temporary initiator device. This transfers the partially regenerated user data from the last XOR data result from the device to the storage array controller;
- f) a REGENERATE command with the intermediate data bit set is sent to a device in domain B. This transfers the regenerate parameter list (containing the other valid device and data extent) from the storage array controller to the device. The partially regenerated user data received in step 5 is sent as the intermediate data;
- g) the device reads the requested data from its own medium. The device performs an XOR operation between the intermediate data and its own data. The resulting XOR data is retained for the next step;
- h) the device becomes a temporary initiator and sends a READ command to the other device included in the regenerate parameter list. That target device transfers the requested data to the temporary initiator. The temporary initiator performs an XOR operation between the XOR data from step 7 and the read data received in this step. The last XOR data result is retained by the temporary initiator. The temporary initiator returns completion status for the REGENERATE command to the storage array controller when this regenerated user data is available;
- i) an XDREAD command is sent from the storage array controller to the device that had been a temporary initiator device in the above steps. This transfers the regenerated user data from the last XOR data result from the device to the storage array controller.

5.1.15.3.3 Rebuild operation

The rebuild operation for this illustration involves five valid devices and the replacement device where three valid devices are in one domain (referred to as domain A) and two valid devices and the replacement device are in the other domain (referred to as domain B). The sequence is as follows (since XOR operands are commutable the XOR order is irrelevant, and the order of steps 2 and 3 is interchangeable):

- a) a REGENERATE command is sent to a device in domain A. This transfers the regenerate parameter list (containing the other two valid devices and data extents) from the storage array controller to the device;
- b) the device reads the requested data from its own medium. The device retains this data for a future XOR operation;
- c) the device becomes a temporary initiator and sends a READ command to another device included in the regenerate parameter list. That device transfers the requested data to the temporary initiator. The temporary initiator performs an XOR operation between the read data from the previous step and the read data received in this step. The resulting XOR data is retained for the next step;
- d) step 3 is repeated until all devices listed in the regenerate parameter list have been accessed. The last XOR data result is retained by the temporary initiator. The temporary initiator returns completion status for the REGENERATE command to the storage array controller when the data is available;
- e) an XDREAD command is sent from the storage array controller to the temporary initiator device. This transfers the partially regenerated data from the device to the storage array controller;
- f) a REBUILD command with the intermediate data bit set is sent to the replacement device in domain B. This transfers the rebuild parameter list (containing the two valid devices and data extents) from the storage array controller to the device. The partially rebuilt data received in step 5 is sent as the intermediate data;
- g) the device becomes a temporary initiator and sends a READ command to a device included in the rebuild parameter list. That target device transfers the requested data to the temporary initiator. The temporary initiator performs an XOR operation between the intermediate data and the read data received in this step. The resulting XOR data is retained for the next step;

- h) the temporary initiator sends a READ command to the other device included in the rebuild parameter list. That device transfers the requested data to the temporary initiator. The temporary initiator performs an XOR operation between the read data from the previous step and the read data received in this step;
- i) the last XOR data result is written to the replacement device's medium, then the device returns completion status for the REBUILD command to the storage array controller.

5.1.15.4 Additional array subsystem considerations

5.1.15.4.0 General

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

5.1.15.4.1 Buffer full status handling

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

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

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

The storage array controller may use command linking to avoid a buffer full condition. For example, a storage array controller supervised update write operation would consist of an XDWRITE command linked to an XDREAD command.

5.1.15.4.2 Access to an inconsistent stripe

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

- a) when an XDWRITE command has been issued and completed, the device containing protected data has been updated but the device containing check data has not. The stripe is inconsistent until the XPWRITE command to the device containing check data returns completion status;
- b) when an XDWRITE EXTENDED command has been issued, the device containing protected data and the device containing check data are updated at different times during the course of the command. The stripe should be treated as inconsistent between the time the storage array controller issues the XDWRITE EXTENDED command and the completion status for the command is received;
- c) any time a regenerate or rebuild operation is in progress for a given stripe, update writes to that stripe should be avoided.

5.1.15.4.3 Error handling considerations

5.1.15.4.3.0 General

If any of the XOR commands end with CHECK CONDITION status and an unrecovered error is indicated, an inconsistent stripe may result. It is the storage array controller's responsibility to identify the failing device and the extent of the failure, then limit access to the inconsistent stripe. In the case of third party XOR operations the failing device may be a device containing protected data or a device containing check data. The storage array controller may identify the failing device from the resulting sense data or, it may access the devices directly to determine the condition of the affected devices. The recovery procedures that the storage array controller implements are not addressed by this standard.

5.1.15.4.3.1 Errors during third party XOR operations

5.1.15.4.3.1.0 General

Third party operations involve the processing of several commands exchanged among three or more devices. For the purposes of this clause, a command that causes the recipient of that command to generate one or more other commands to another device (or devices) is referred to as a primary command and the recipient of a primary command is referred to as a primary target. All commands generated by a primary target (based on the receipt of a primary command) are referred to as secondary commands and are sent to a secondary target or secondary targets. The definitions of primary command, primary target, secondary command, and secondary target are temporary, and for this clause only, and they should not be associated with the more general "SCSI Primary Command" or "SCSI Target" usage.

The primary target of an XDWRITE EXTENDED primary command generates an XPWRITE secondary command; the primary target of a REBUILD primary command generates one or more READ secondary commands. The primary command shall not be completed until the primary target is prepared to report the completion status or implied completion status of the secondary commands. Two classes of exception conditions may occur during these commands. One class consists of those resulting directly from the primary command. The other class consists of those resulting from a secondary command. Either or both of these classes of exception may occur during a third party operation.

5.1.15.4.3.1.1 Primary errors – Errors resulting directly from the primary command

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

- a) terminate the primary command with CHECK CONDITION status;
- b) build sense data according to the exception condition.

5.1.15.4.3.1.2 Secondary errors – Errors resulting from the secondary command

The second class of errors consists of exception conditions resulting from the failure of a secondary command. The sense data for such errors shall be passed to the initiator of the primary command in the additional sense bytes field of the sense data.

If the primary target detects the exception (i.e., by some means other than receiving CHECK CONDITION status from the secondary target), it shall:

- a) terminate the primary command with CHECK CONDITION status;
- b) set the sense key to ABORTED COMMAND if there are no primary errors to report. Otherwise, the sense key shall be set according to the primary error;
- c) set the first byte of the command specific information field of the sense data to the starting byte number, relative to the first byte of sense data, of an area that contains the primary target's sense data for the secondary error. A zero value in this byte indicates no secondary error has been detected by the primary target. The secondary sense data shall be built in the standard sense data format as defined for the REQUEST SENSE command;
- d) in the case of a REBUILD or REGENERATE primary command, set the third byte of the command specific information field of the sense data to an index value indicating the target identifier of the failing secondary target. This value shall be an index into the source descriptor entries of the parameter data of the primary command, and shall point to the entry containing the target identifier of the failing device; 0 points to the first entry, 1 points to the second entry, etc. This byte shall be ignored if the primary command is not a REBUILD or REGENERATE.

If the secondary target detects the exception, the primary target receives CHECK CONDITION status from the secondary target. The primary target shall recover the sense data associated with the exception condition, clear any exception conditions associated with the CHECK CONDITION status and shall:

- a) terminate the primary command with CHECK CONDITION status;
- b) set the sense key to ABORTED COMMAND if there are no primary errors to report. Otherwise, the sense key shall be set according to the primary error;
- c) set the second byte of the command specific information field of the sense data to the starting byte number, relative to the first byte of sense data, of an area that contains (unchanged) the secondary target's status byte followed by its sense data. A zero value in this byte indicates no secondary error has been reported by the secondary target;
- d) in the case of a REBUILD or REGENERATE (primary) command, set the third byte of the command specific information field of the sense data to an index value indicating the target identifier of the failing secondary target. This value shall be an index into the source descriptor entries of the parameter data of the primary command and shall point to the entry containing the target identifier of the failing device. 0 points to the first entry, 1 points to the second entry, etc. This byte is invalid and shall be ignored if the primary command is not a REBUILD or REGENERATE.

For a given primary command, if errors are generated by more than one secondary command, the sense data shall contain error information for the secondary error first obtained by the primary target.

Since, for secondary errors, the sense key is set to ABORTED COMMAND only if there are no primary errors to report (see item b) above), the first and second bytes of the command specific information field should be checked, even when the sense key is a value other than ABORTED COMMAND, to determine if any secondary errors have occurred.

NOTE 1 All three of the above error types might occur during the same third party operation. If this happens, there are three unique pieces of error information contained in the sense data: one for the primary error (starting at byte 0), and two for the secondary errors (in the additional sense bytes).

5.1.15.4.4 XOR data retention requirements

The target shall retain XOR data while awaiting retrieval by an XDREAD command until performing one of the following events: a matching XDREAD command, TARGET RESET, power cycle, CLEAR TASK SET, ABORT TASK if the task matches the pending XDREAD, ABORT TASK SET.

5.2 Model for optical memory block devices

5.2.0 General

An optical memory block device is a logical unit that can potentially support a variety of optical media, (for example, read-only, write-once, erasable or reversible). In several respects, an optical memory block device is similar to a direct-access block device. However, optical memory block devices may offer features that are not available with other logical units, including large capacity removable media.

These logical units often require the functions that are not found in direct-access block devices such as logical block update, pre-erasure before writing, or scanning for blank medium and twelve-byte command descriptor blocks. This standard includes specific device types for write-once and CD-ROM block devices that also use optical media, but are not capable of supporting several types of optical media. A logical unit that uses write-once media can be an optical memory block device. Logical units that use read-only media can be optical memory block devices. However, logical units using CD-ROM media have certain unique characteristics and should not be implemented as optical memory block devices.

A model of optical memory block devices is complicated by the nature of one of its potential advantages, that it can support media that has different characteristics. There are three types of optical media in general use, read-only, write-once and reversible. Read-only media are used for publishing applications requiring dissemination of large amounts of data since the data may be replicated on a disk at low cost. Write-once media are used in applications that have large backup or archiving requirements. It is also used in applications that need large amounts of on-line reference information. Reversible media is used in applications that need large amounts of temporary storage (for example, a graphics workstation) and can take advantage of removable media. In some applications, reversible media devices are used instead of direct-access block devices.

Reversible media usually need to be reversed (erased, blanked) before new data can be written. In such cases an erase operation is required before data can be written. Some optical memory block devices perform this erase operation implicit with each write operation that impacts the data throughput. Some block devices can perform the erase separately. The ERASE command may be used to erase areas of the medium with a corresponding increase in data throughput on subsequent write operations. Products using optical media should not be implemented as direct-access block devices, due to the overhead penalty on performance from the emulation and the lack of support in direct-access block devices to take advantage of the features specifically available with optical memory block devices.

The type of medium supported by the block device and the type of medium currently loaded may be determined by examining the MODE SENSE data. One unique feature of optical memory block devices is support of media with mixed types (for example, media with read-only and write-once areas). The INQUIRY command informs the application client that the logical unit is an optical memory block device. The application client should then determine the medium type from the MODE SENSE data. The application client needs to be cognizant of the medium type since the logical unit's characteristics may change when the media are changed.

Write-once media can have valid data written to a logical block once. This is an important feature where audit trails and permanent archives are needed. Optical memory block devices supporting write-once media may have the ability to update a logical block, preserving the previous generation of data. These logical units may provide a means to recover the previous data through use of commands that allow read access to the different generations of data that are stored at the same logical block address.

An important requirement in dealing with optical media is determining if logical blocks contain written data and or if they are blank. A blank logical block is one that is properly initialized so that data subsequently written to it can be recovered. The logical blocks may have a flag associated with each that indicates whether they have been written or not.

Strategies used to manage write once and erasable media may depend on being able to determine the boundary between written and blank areas of the medium. The MEDIUM SCAN command is useful in finding blank areas for subsequent write operations.

5.2.1 Defect management

Defect management may be performed on logical blocks by updating in a manner similar to that used by direct-access block devices with the REASSIGN BLOCKS command. The advantage of using the updating (that is not supported by direct-access block devices) is access to the previous data.

The update operation assigns an alternate physical block to the logical block while simultaneously writing the data to the block. Commands are provided to allow the recovery of previous generations of updated blocks.

Defect management on optical-memory block devices may be vendor-specific. However, there are standards for some types of optical-memory media that specify defect management techniques. These standards may supersede the requirements pertaining to error and defect reporting in this standard.

5.2.2 Error reporting

If any of the following conditions occur during the execution of a command (see Table 2), the device server shall return CHECK CONDITION status and the appropriate sense key shall be set with the appropriate additional sense code for the condition. The following list illustrates some error conditions and the applicable sense keys. The list does not provide an exhaustive enumeration of all conditions that may cause the CHECK CONDITION status.

Table 2 – Sample error conditions

Condition	Sense key
Invalid logical block address	ILLEGAL REQUEST
Unsupported option requested	ILLEGAL REQUEST
Logical unit reset 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
Overrun or other error that might be resolved by repeating the command	ABORTED COMMAND
Attempt to write on write protected medium	DATA PROTECT
Attempt to read a blank or previously unwritten block	BLANK CHECK
Attempt to write a previously written block and blank checking is enabled	BLANK CHECK
Attempt to write on read-only medium	DATA PROTECT

In the case of an invalid logical block address, the sense data information field shall be set to the logical block address of the first invalid address.

In the case of an attempt to read a blank or previously unwritten block, the sense data information field shall be set to the logical block address of the first blank block encountered. The data read up to that block shall be transferred.

In the case of an attempt to write a previously written block when blank checking is enabled, the sense information field shall be set to the logical block address of the first non-blank block encountered.

5.3 Model for write-once block devices

5.3.0 General

The model for the write-once block device is a variation on the optical memory model. Most of the aspects of a write-once block device are similar to optical memory block devices. The differences are summarized in this subclause.

5.3.1 Logical blocks

Data may be written to a logical block only once. A subsequent write to a logical block already written may or may not be corrupted, depending on the implementation. Write-once physical media is non-volatile.

SCSI write-once block devices are intended to be archival in nature. Data at a logical block address is not expected to change once it is written. The update commands are not recommended for this device type. Logical units that require the update function should use the optical memory device type.

Block devices may be able to determine the state of a logical block prior to access. These block devices can determine whether a block is blank or written. This is useful in detecting previously written blocks and preventing a destructive overwrite. This is also useful in finding blank areas for later writing. The MEDIUM SCAN command may be used to find blank and written areas prior to WRITE and READ commands.

5.3.2 Initialization

The FORMAT UNIT command is not used by write-once block devices. Write-once media is shipped pre-formatted by the manufacturer and is ready for use when mounted.

5.3.3 Physical medium defects

The raw defect rate may be higher for optical medium than for magnetic medium. Data may or may not be recovered through the use of sophisticated error correction algorithms. The level of error correction used for data recovery may be selectable. However, write-once block devices may have a minimum level that is always used and is not changeable through the error recovery mode parameter. Control of the error correction algorithms and level of correction is vendor-specific.

Defect management on write-once block devices may be vendor-specific. However, there are standards for some types of write-once media that specify defect management techniques. These standards, may supersede the implementation requirements pertaining to error and defect reporting in this standard.

5.3.4 Error reporting

If any of the following conditions occur during the execution of a command (see Table 3) the device server shall return CHECK CONDITION status and the appropriate sense key shall be set with the additional sense code for the condition. The following list illustrates some error conditions and the applicable sense keys. The list does not provide an exhaustive enumeration of all conditions that may cause the CHECK CONDITION status.

Table 3 – Sample error conditions

Condition	Sense key
Invalid logical block address	ILLEGAL REQUEST
Unsupported option requested	ILLEGAL REQUEST
Logical unit reset 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
Overrun or other error that might be resolved by repeating the command	ABORTED COMMAND
Attempt to write on write protected medium	DATA PROTECT
Attempt to read a blank or previously unwritten block	BLANK CHECK
Attempt to write a previously written block and blank checking is enabled	BLANK CHECK

In the case of an invalid logical block address, the sense data information field shall be set to the logical block address of the first invalid address.

In the case of an attempt to read a blank or previously unwritten block, the sense data information field shall be set to the logical block address of the first blank block encountered. The data read up to that block shall be transferred.

In the case of an attempt to write a previously written block and blank checking is enabled, the sense information field shall be set to the logical block address of the first non-blank block encountered.

6 Commands for block devices

6.1 Commands for direct-access block devices overview

6.1.1 Commands for direct-access block devices

The commands for direct-access block devices shall be as shown in Table 4.

Table 4 – Commands for direct-access block devices

Command name	Operation code	Type	Subclause
CHANGE DEFINITION	40h	O	ISO/IEC 14776-311, SPC
COMPARE	39h	O	ISO/IEC 14776-311, SPC
COPY	18h	O	ISO/IEC 14776-311, SPC
COPY AND VERIFY	3Ah	O	ISO/IEC 14776-311, SPC
FORMAT UNIT	04h	M	6.1.2
INQUIRY	12h	M	ISO/IEC 14776-311, SPC
LOCK-UNLOCK CACHE	36h	O	6.1.3
LOG SELECT	4Ch	O	ISO/IEC 14776-311, SPC
LOG SENSE	4Dh	O	ISO/IEC 14776-311, SPC
MODE SELECT(6)	15h	O	ISO/IEC 14776-311, SPC
MODE SELECT(10)	55h	O	ISO/IEC 14776-311, SPC
MODE SENSE(6)	1Ah	O	ISO/IEC 14776-311, SPC
MODE SENSE(10)	5Ah	O	ISO/IEC 14776-311, SPC
MOVE MEDIUM	A7h	O	SMC
Obsolete	01h	OB	3.3.4
Obsolete	31h	OB	3.3.4
Obsolete	30h	OB	3.3.4
Obsolete	32h	OB	3.3.4
Obsolete	0Bh	OB	3.3.4
PERSISTENT RESERVE IN	5Eh	O ¹	ISO/IEC 14776-311, SPC
PERSISTENT RESERVE OUT	5Fh	O ¹	ISO/IEC 14776-311, SPC
PRE-FETCH	34h	O	6.1.4
PREVENT-ALLOW MEDIUM REMOVAL	1Eh	O	ISO/IEC 14776-311, SPC
READ(6)	08h	M	6.1.5
READ(10)	28h	M	6.1.6
READ(12)	A8h	O	6.2.4
READ BUFFER	3Ch	O	ISO/IEC 14776-311, SPC
READ CAPACITY	25h	M	6.1.7
READ DEFECT DATA(10)	37h	O	6.1.8
READ DEFECT DATA(12)	B7h	O	6.2.5
READ ELEMENT STATUS	B4h	O	SMC
READ LONG	3Eh	O	6.1.9
REASSIGN BLOCKS	07h	O	6.1.10
REBUILD	81h	O	6.1.11
RECEIVE DIAGNOSTIC RESULTS	1Ch	O	ISO/IEC 14776-311, SPC

Table 4 (continued)

Command name	Operation code	Type	Subclause
REGENERATE	82h	O	6.1.12
RELEASE(6)	17h	O ²	ISO/IEC 14776-311, SPC
RELEASE(10)	57h	M	ISO/IEC 14776-311, SPC
REPORT LUNS	A0h	O	ISO/IEC 14776-311, SPC
REQUEST SENSE	03h	M	ISO/IEC 14776-311, SPC
RESERVE(6)	16h	O ²	ISO/IEC 14776-311, SPC
RESERVE(10)	56h	M	ISO/IEC 14776-311, SPC
SEEK(10)	2Bh	O	6.1.13
SEND DIAGNOSTIC	1Dh	M	ISO/IEC 14776-311, SPC
SET LIMITS(10)	33h	O	6.1.14
SET LIMITS(12)	B3h	O	6.2.8
START STOP UNIT	1Bh	O	6.1.15
SYNCHRONIZE CACHE	35h	O	6.1.16
TEST UNIT READY	00h	M	ISO/IEC 14776-311, SPC
VERIFY	2Fh	O	6.1.17
WRITE(6)	0Ah	O	6.1.18
WRITE(10)	2Ah	O	6.1.19
WRITE(12)	AAh	O	6.2.13
WRITE AND VERIFY	2Eh	O	6.1.20
WRITE BUFFER	3Bh	O	ISO/IEC 14776-311, SPC
WRITE LONG	3Fh	O	6.1.21
WRITE SAME	41h	O	6.1.22
XDREAD	52h	O	6.1.23
XDWRITE	50h	O	6.1.24
XDWRITE EXTENDED	80h	O	6.1.25
XPWRITE	51h	O	6.1.26
Key M = Command implementation is mandatory O = Command implementation is optional OB = Obsolete SPC = SCSI-3 Primary Commands SMC = SCSI-3 Medium Changer Command Set			
NOTE 1 Optional PERSISTENT RESERVE Commands, if implemented, shall both be implemented as a group. NOTE 2 Optional RELEASE(6) and RESERVE(6) Commands, if implemented, shall both be implemented as a group. NOTE 3 The following operation codes are vendor-specific: 02h, 05h, 06h, 09h, 0Ch, 0Dh, 0Eh, 0Fh, 10h, 13h, 14h, 19h, 20h, 21h, 22h, 23h, 24h, 26h, 27h, 29h, 2Ch, 2Dh and C0h through FFh. All remaining operation codes for direct-access block devices are reserved for future standardization.			

6.1.2 FORMAT UNIT command

6.1.2.0 General

The FORMAT UNIT command (see Table 5) formats the medium into application client addressable logical blocks according to the application options defined by the client. In addition, the medium may be certified and control structures may be created for the management of the medium and defects. The degree to which the medium is altered by this command is vendor-specific.

Table 5 – FORMAT UNIT command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (04h)							
1	Reserved			FMTDATA	CMPLST	DEFECT LIST FORMAT		
2	VENDOR-SPECIFIC							
3	(MSB)							
4	INTERLEAVE (LSB)							
5	CONTROL							

The simplest mandatory form of the FORMAT UNIT command (with no format 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. Two additional mandatory forms of this command increase the application client's control over defect management. Several optional forms of this command further increase the application client's control over defect management, by allowing the application client to specify

- defect list(s) to be used,
- defect locations (in several formats),
- that logical unit certification be enabled, and
- exception handling in the event that defect lists are not accessible.

If the logical unit is reserved, a reservation conflict shall occur when a FORMAT UNIT command is interpreted from an initiator other than the one holding a logical unit reservation. The command shall be rejected with RESERVATION CONFLICT status if the reservation conflict is due to a logical unit reservation, or any extent reservation, from any application client, is active for the specified logical unit.

During the format operation, the device server shall respond to commands as follows:

- a) in response to all commands except REQUEST SENSE and INQUIRY, the device server shall return CHECK CONDITION status unless a reservation conflict exists, in that case RESERVATION CONFLICT status shall be returned;
- b) in response to the INQUIRY command, the device server shall respond as commanded;
- c) in response to the REQUEST SENSE command, unless an error has occurred, the device server shall return a sense key of NOT READY and an additional sense code of LOGICAL UNIT NOT READY FORMAT IN PROGRESS, with the sense key specific bytes set for progress indication (as described in *ISO/IEC 14776-311, SCSI-3 Primary Commands*). Refer to *ISO/IEC 14776-311, SCSI-3 Primary Commands* for a description of deferred error handling that may occur during the format operation.

NOTE 1 The MODE SELECT parameters (if any) should be set prior to issuing the FORMAT UNIT command.

During the execution of the FORMAT UNIT command, the device server may perform a medium defect management algorithm (that may be controlled by the application client, using