# Working Draft American National Standard

# Project T10/BSR INCITS 514

Revision 35
07 December 2012

## Information technology - SCSI Block Commands – 3 (SBC-3)

This is an internal working document of T10, a Technical Committee of Accredited Standards Committee INCITS (International Committee for Information Technology Standards). As such this is not a completed standard and has not been approved. The contents may be modified by the T10 Technical Committee. The contents are actively being modified by T10. This document is made available for review and comment only.



T10 Technical Editor:

Mark Evans
Western Digital Corporation
5863 Rue Ferrari
San Jose, CA 95138
USA

Telephone: 408-363-5257
Email: mark.evans@wdc.com

**Reference number**
**ISO/IEC 14776-323:201x**
**BSR INCITS 514:201x**

## Points of contact

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

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

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

**T10 Vice-Chair**
Mark S. Evans
Western Digital Corporation
5863 Rue Ferrari
San Jose, CA 95138
USA

Telephone: (408) 363-5257
Email: mark.evans@wdc.com

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

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

**INCITS Secretariat**
1101 K Street, NW
Suite 610
Washington, DC 20005
USA

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

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

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

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

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

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

American National Standard
for Information Technology

# SCSI Block Commands – 3 (SBC-3)

Secretariat
**Information Technology Industry Council**

Approved mm.dd.yy

American National Standards Institute, Inc.

## Abstract

This standard specifies the functional requirements for the SCSI Block Commands - 3 (SBC-3) command set. SBC-3 permits SCSI block logical units such as rigid disks to attach to computers and provides the definition for their use.

This standard maintains a high degree of compatibility with the SCSI Block Commands (SBC-2) command set, INCITS 405-2005, and while providing additional functions, is not intended to require changes to presently installed devices or existing software.

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer. Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that effort be made towards their resolution.

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

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

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

**CAUTION:** The developers of this standard have requested that holders of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard. As of the date of publication of this standard, following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Published by
**American National Standards Institute**
**11 W. 42nd Street, New York, New York 10036**



Printed in the United States of America

## Revision History

This revision history is not part of American National Standard INCITS 1799:200x.

## 1 Revision 0 (09 September 2005)

Revision 0 of SBC-3 is substantially equal to revision 16 of SBC-2. The only differences arise from changes made in SBC-2 discovered during the ISO process. Those changes include:

a) Changed idle condition timer to standby condition timer in item c) of subclause 4.15.1.
b) Changed 2 Gigabytes to 1 GiB and 2 Terabytes to 2 TiB in two places in note 10 in subclause 5.5. The 2 was change to a 1 because there are 21 bits in the LOGICAL BLOCK ADDRESS field (i.e., 1F_FFFF is 2 097 151 that * 512 is 1 073 741 312 which is 1 GiB).

Removed the CORRCT bit from the WRITE LONG (16) CDB as it was never supposed to be added to this command. It's not in SCSI-2 or SBC for WRITE LONG (10) and 03-383r1 did not ask for it. It showed up in sbc2r11.

## 2 Revision 1 (16 September 2005)

a) Incorporated the following proposals:
   A) 04-198r5 - Background medium Scan;
   B) 04-371r2 - SPC-4: Enable Background Operation Error Reporting Bit; and
   C) 05-101r1 - SBC-2 Validation of Protection Information.

## 3 Revision 2 (22 September 2005)

a) Incorporated the following proposals:
   A) 05-299r1 - Correct Log Page Format Tables in SPC-4, SBC-3, & SAS-2; and
   B) 05-313r0 - SBC-3: Change to background medium scan.

## 4 Revision 3 (16 November 2005)

a) Incorporated the following proposals:
   A) 05-156r7 - SBC-3, SPC-4: Application ownership of protection information Reference Tag;
   B) 05-317r3 - SMC-3, SPC-4, SBC-3, and SSC-3: Remove Attached medium Changer model; and
   C) 05-374r2 - SBC-3: SPC-4: Disabling Reassign on Write Long Logical Blocks.

## 5 Revision 4 (10 February 2006)

a) Incorporated the following proposals:
   A) 05-157r9 - SPC Security Commands proposal (in an E-mail it was pointed out that the SERCURITY PROTOCOL IN and SECURITY PROTOCOL OUT commands need to be added to the SBC-3 commands list table);
   B) 05-340r3 - SBC-3 SPC-4 Background scan additions;
   C) 05-368r2 - SPC-4 SBC-3 SMC-3 Allow more commands through Write Exclusive reservations; and
   D) 05-383r4 - SPC-4: Deferred microcode downloads.

In addition the following editorial corrections were received from E-mail were incorporated:

a) the initiator control (IC) enable bit description had the SIZE bit polarity backwards;
b) changed filed to field;
c) WRITE SAME (32) was placed into the list of medium access commands in the Protection types overview subclause;
d) the field name P_TYPEABLE in table 19 (**FMTPINFO** bit, **RTO_REQ** bit, and **PROTECTION FIELD USAGE** field) footnote d was changed to P_TYPE; and

e) the field name DATA BLOCK GUARD in table 80 (LBDATA bit and PBDATA bit) in row 0 0 was changed to LOGICAL BLOCK GUARD.

## 6 Revision 5 (11 May 2006)

a) Incorporated the following proposals:
   A) 06-248r1 - Proposal to remove the PREVENT ALLOW MEDIUM REMOVAL (PAMR) command from SPC-4

In addition the following editorial corrections were received from E-mail were incorporated:

a) The following sentence occurs on SBC-3 rev. 3, page 122, first paragraph under table 110, last sentence of paragraph; and also on page 124, first paragraph under table 111, last sentence of paragraph:
   “To determine the number of blocks at which the logical unit is currently formatted, the application client shall use the READ CAPACITY command (see 5.11) rather than the MODE SELECT command.”
   In both cases, the “MODE SELECT” needs to be replaced with “MODE SENSE”. To use the “select” operation is nonsensical if the purpose is to discover the current block size of the disk drive.

## 7 Revision 6 (24 July 2006)

a) Incorporated the following proposals:
   A) 06-259r1 - SAM-4, et al.: making linked commands obsolete;
   B) 06-274r1 - SPC-4 SBC-3 REQUEST SENSE and Stopped power condition; and
   C) 06-323r1 - SAM-4 SPC-4 et al Multiple service delivery subsystem editorial tweaks.

## 8 Revision 7 (22 September 2006)

a) Incorporated the following proposals:
   A) 06-034r5 - SBC-3 Physical blocks; and
   B) 06-350r0 - SPC-4/SBC-3: Power conditions state machine clarification.

In addition the following editorial corrections were received from E-mail were incorporated:

a) In the Background Scan Results log page (section 6.2.2) in table 101 on page 119, there seems to be a missing “scan” in the row for code 08h. It currently reads:
   “Background medium halted, waiting...”.
   GAH: Agreed, it should read “Background medium scan halted, waiting...”. I note that the word “medium” is not present in the descriptions of other code rows, maybe that word should be removed for consistency.
   GOP: The word “medium” was added to all the descriptions of other code rows to make it consistent with the rest of the clause.
b) In “Background medium scan parameter format” for parameter code 0001h to 0800h shown in table 102 the “accumulated power on minutes” field is defined on page 121 as “The ACCUMULATED POWER ON MINUTES field indicates the number of minutes the device server has been powered on since manufacturing.” That is the same definition of the same named field found in the “Background medium scan status parameter format” (for parameter code 0) shown in table 99. Shouldn't the description associated with table 102 be referencing when the error associated with that parameter number was logged [similar to the way the self test log page outputs its results]?
   GAH: I agree. It should say “The ACCUMULATED POWER ON MINUTES field indicates the number of minutes the device server has been powered on since manufacturing at the time the background scan error occurred.”

## 9 Revision 8 (18 January 2007)

Fixed the byte numbering in the READ CAPACITY (16) parameter data table in 5.16.2.

Based on the following E-mail from Mark Evans pointing out error in the incorporation of proposal 05-340r3 changes were made to table 111 and table 114 and a description of the DS bit and the SPF bit were placed under table 108:

> While going through SBC-3, I see that table 102 (parameter control bits for Background Scanning Status log parameter) looks goofy. I think it should look like table 194 in SAS. I'm making that change in our internal document and recommend that you make the change in SBC.

Incorporated the following proposals:

a) 06-479r1 - Mandate CAPACITY DATA HAS CHANGED unit attention; and
b) 06-393r3 - On-disk bitmap support.

## 10 Revision 8a (18 January 2007)

The ORWRITE CDB table title was fixed and the ORWRITE operation code in the CDB was corrected.

## 11 Revision 9 (22 March 2007)

Based on the following 1/23/2007 E-mail from Mark Evans pointing out a duplicate entry between subclause 4.17 Protection information model and subclause 5.31 WRITE AND VERIFY (10) command. The duplicate wording that needs to be deleted is from 5.31 WRITE AND VERIFY (10) command and is:

> If the P_TYPE bit is set to one in the READ CAPACITY (16) parameter data (see 5.13), the device server shall terminate this command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE.
> During the investigation of this more duplicate wording was found in subclause 5.38 WRITE SAME (16) command. This duplication wording that needs to be deleted is:
> If the P_TYPE bit is set to one in the READ CAPACITY (16) parameter data (see 5.13), the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE
> In both cases the duplicate wording should have been removed as part of T10/05-156 revision 7.

Based on the following 1/26/2007 E-mail from Rob Elliott editorial changes were made as recommended in the e-mail. The only recommend change not made was to change ORWRITE to ORWRITE (16). Instead all the ORWRITE (16)s were changed to ORWRITE.

1) On page 55, this text needs small caps and the table references need to be fixed:
   The device server shall:
   a) check protection information read from the medium based on the ORPROTECT field as described in tableyy2; and
   b) check protection information transferred from the data-out buffer based on the ORPROTECT field as described in table yy3.
2) In table 3 (reservations), on page 29, and on page 30, ORWRITE s/b ORWRITE (16)
3) In the table of tables on page xiii, the prevent field doesn't show up in small caps like the others.
4) On page 34 and 35, several of the equations are truncated on the left and right. The reason is the font size grew - the Equation format or character format is messed up on those lines.

Based on the following 2/8/2007 emails from Dave Peterson and Rob Elliott pointing out a badly worded sentence that sentence was corrected.

> Dave Peterson:
> Sentence in question:
> Subclause 5.2.2.3 first paragraph, first sentence on physical page 52:
> “When the SI bit is set to one, the device server need not write the initialization pattern over the header and other header and other parts of the medium not previously accessible to the application client.”
> The extra “...and other header...” appears odd. Please clarify.
> Rob Elliott:
> I agree with David's fix.

Based on the following email from Doug Gilbert received on 2/23/2007 I added in cross-references to the footnote in table 84. The cross-references were added to all the yes terms in the 3rd column.

> George, you should delete the spurious “and other header” words from the SI bit paragraph SBC-3. That was broken in sbc2r12 while incorporating editor's meeting comments to fix a parenthetical expression without an e.g. or i.e.
> In section 5.35 on WRITE LONG(10) there is table 84 which has a footnote “a”. It is hanging because there
> seems nothing in the main table it refers to. My guess is that it refers to the third column header which starts “More than one logical block...”.

Incorporated the following proposals:

a) 07-110r0 - Update Block Limits VPD Page for ORWRITE; and
b) 07-113r0 - Maximum transfer sizes for XPWRITE XDWRITE XDREAD PRE-FETCH.

## 12 Revision 10 (17 May 2007)

Incorporated the following proposals:

a) 07-203r0 - SBC-3 SPC-4 Block Device Characteristics VPD page and medium rotation rate field; and
b) 07-208r0 - SBC-3 Rename field in READ CAPACITY (16) parameter data.

## 13 Revision 11 (19 July 2007)

All mode page format tables have been updated to include the SubPage Format (SPF) bit.

Incorporated the following proposals:

a) 07-257r1 - Prohibited needed as a keyword in SPC-4;
b) 07-271r1 - SBC-3, Clarifications for Background Scan Results log page;
c) 07-281r1 - SPC-4, SBC-3, #Except INQUIRY, REPORT LUNS, and REQUEST SENSE#; and
d) 07-302r1 - SBC-3 WRITE LONG Additional Sense code option to support SAT-2.

Based on the following email from Rob Elliott received on 5/16//2007:

1. SBC-3 uses both “medium access commands” (from changes made in the DIF area) and “medium access commands.” SBC-2 only used “medium access commands.”

I changed all the “medium access commands” to “medium access commands”.

As a result of 07-271 references to generic fields (e.g., OPERATION CODE field, PAGE CODE field, PAGE LENGTH field) were added under the tables where those fields were defined.

## 14 Revision 12 (11 November 2007)

Incorporated the following proposals:

a) 07-472r0 - Reporting nominal form factor;
b) 07-447r1 - Read-Write Error Recovery clarifications; and
c) 07-481r1 - Mention that DIF equals protection information.

Made editorial changes based on the following email received from Rob Elliott on 7/24/2007

> You added wording like this to many sections:
> “The OPERATION CODE field is defined in SPC-4 shall be set to the value defined in table 72." but “shall” needs to be “and shall”.
> I see a few that are broken, though: After table 81, it points to table 72. After table 91, it points to table 90.
> The wording for SPF bit in the mode pages:
> “A SubPage Format (SPF) bit set to zero indicates that the page_0 mode page format is being used (see SPC-4).” should probably be:

"The SubPage Format (SPF) bit is defined in SPC-4 and shall be set to the value defined in table xx."xx."" (probably combined with the PAGE CODE sentence in all cases)
Table 102, bytes 4-19 - parameters s/b parameter (there's only one)
Table 104 - add (MSB)/(LSB) on the last field
Table 112 - in the sentence after this table, the period is on the wrong line.
Table 113, 115 - change 3h to 03h
Editor's note 1: Either that proposal or the editor deleted the wrong line from SBC-2 when obsoleting linked commands. It should just be "CONDITION MET" not "INTERMEDIATE-CONDITION MET". See SBC-2's original wording - the LINK bit set to zero sentences are the ones that should have remained.
Added the control byte reference to SAM-4 for all the CDBs.

Made editorial changes as noted in the following email received from Gerry Houlder on 11/06/2007:

This inconsistent behavior was noted by an engineer at Seagate while reviewing Protection Information behavior in SBC-3. It would seem to be more appropriate to call out 05/24 sense instead of 05/20 to be consistent with products that report 05/24 for reserved fields.

## 15 Revision 13 (24 January 2008)

Made all binary and hexadecimal numbers have a consistent format, separating groups of four digits with underscores. Changed the conventions clause (see clause 3.6) to define this format.

Incorporated the following proposals:

a) 07-451r1 - WRITE LONG COR_DIS and WR_UNCOR interaction;
b) 07-454r5 - Capability based Command Security;
c) 07-459r4 - Unit attention condition queuing;
d) 08-041r1 - Use period as decimal separator in T10 standards;

Made editorial changes as noted in the following email received from Rob Elliott on 01/03/2008:

- add the subpage code column to the table of log page codes
- add 00h/FFh as Supported Log Pages and Subpages to the table of log page codes
- add 01h-3Eh/FFh as Supported Subpages to the table of log page codes
- add 18h/00h-3Eh as Protocol Specific Port log pages to the table of log page codes
- add the DS bit and SPF bit (0b) to byte 0 of any log page definitions
- add the SUBPAGE CODE field (00h) to byte 1 of any log page definitions
- make sure the command set table agrees with the SPC-4 annex about the page names, reserved and vendor-specific ranges, etc.

## 16 Revision 14 (20 March 2008)

Incorporated the following proposals:

a) 08-139r1 - START STOP UNIT command additions.

Made the editorial change in 4.21 based on an email from Fred Knight: changed "idle" to "standby" in list item (C) in the bulleted list for how to process a REQUEST SENSE command while in the standby power condition.

Updated the definition of the PREVENT ALLOW MEDIUM REMOVAL command to be as proposed in 05-317r3 based on emails from Rob Elliott.

Made other editorial changes as denoted by change bars, including:

- added definitions for device server, I_T nexus, logical unit, SCSI target device, and unit attention condition
- added requirement that, if the medium in a direct-access block device is removable, and the medium is removed, then the device server shall establish a unit attention condition

Made other minor editorial and formatting changes not indicated by change bars.

## 17 Revision 15 (15 May 2008)

Incorporated the following proposals:

a) 08-156r2 -Non-volatile cache becoming volatile.

Made other minor editorial and formatting changes not indicated by change bars.

## 18 Revision 16 (25 August 2008)

Incorporated the following proposals:

a) 08-116r3 - SBC-3 SPC-4: Protection Type 3 Reference Tag Clarification.

Based on an email exchange between Mark Evans and Rob Elliott, changed the following sentence in 4.9 Medium defects from, “During a self-test operation (see SPC-4), the device server shall ignore pseudo unrecoverable errors with correction disabled and shall process the pseudo unrecoverable errors with correction disabled.” to “During a self-test operation (see SPC-4), the device server shall ignore pseudo unrecoverable errors with correction disabled and shall process the pseudo unrecoverable errors with correction enabled.”

The conventions subclause was updated to include the latest descriptions of the conventions for lettered and numbered lists.

The definition for how a range of numeric values are represented in this standard was added, and the attempt was made to find all instances of “through”, “-”, and other representations where they were used to represent a range of numbers and replaced each instance with “to” to conform to this definition.

Other minor editorial and formatting changes were made that are not indicated by change bars.

## 19 Revision 17 (17 November 2008)

Incorporated the following proposals:

a) 08-145r3 - Capability-based Command Security (CbCS) [the rewrite];
b) 08-192r1 - SBC-3 Model for encrypting disk drives; and
c) 08-450r1 - Add Restricted keyword to the Style Guide.

Minor editorial corrections were made, several based on email input, which are indicated by change bars.

Other minor editorial and formatting changes were made that are not indicated by change bars.

## 20 Revision 18 (23 February 2009)

Incorporated the following proposals:

a) 08-396r3 - SPC-4/SBC-3: Reporting support for all DIF types
b) 09-007r0 - SBC-3: REASSIGN BLOCKS and CbCS;
c) 09-054r1 - SPC-4/SBC-3/SPL: Adding more idle power options;
d) 08-356r5 - SBC-3: Thin Provisioning Commands; and
e) 08-415r4 - SBC-3/SPC-4: Adding a Protection Information Interval.

Based on an email exchange with Dan Colegrove and Rob Elliot, clarified the table defining the values in the REASSIGN STATUS field (see table 155) in the Background Scan Results log page (see 6.4.2) as was intended by proposal 05-340r3.

Minor editorial corrections were made, several based on email input, which are indicated by change bars.

Other editorial and formatting changes were made that may not be indicated by change bars.

## 21 Revision 19 (29 May 2009)

Incorporated the following proposals:

a) 09-038r3 - SBC-3: More background scan clean-up;
b) 09-069r0 - SBC-3: Minor consistence adjustment for XOR commands; and
c) 09-153r1 - SBC-3: Thin Provisioning per LBA cleanup and granularity.

09-306r1 - SBC-3: GET LBA STATUS w/normalized Allocation Length definition

Minor editorial corrections were made, several based on email input, which are indicated by change bars.

Other editorial and formatting changes were made that may not be indicated by change bars.

## 22 Revision 20 (10 September 2009)

Incorporated the following proposals:

a) 08-341r6 - SBC-3 Thin Provisioning GET LBA STATUS Command
b) 09-088r2 - Allow vendor specific log parameters in Log page 15h;
c) 09-160r2 - Protection Information settings during capacity change;
d) 09-202r1 - Obsolete commands and pages in SBC-3; and
e) 09-011r8 - SBC-3 Thin Provisioning Thresholds.

Where they were not already, the tables defining the codes for the diagnostic parameters, log parameters, mode parameters, and vital product data (VPD) parameters were moved to the beginning of their respective clauses, and the parameters were placed in alphabetical order.

Minor editorial corrections were made, several based on email input, which are indicated by change bars.

Other editorial and formatting changes were made that may not be indicated by change bars.

## 23 Revision 21 (25 November 2009)

Incorporated the following proposals:

a) 09-004r4 - SBC-3: Host alignment detection;
b) 09-088r2 - SBC-3: GET LBA STATUS w/ normalized Allocation Length definition;
c) 09-322r2 - SBC-3: Add log page for solid state drive (SSD);
d) 09-365r1 - SBC-3: Incorrect description of LOWIR bit effect in table 118;
e) 09-372r1 - SBC-3: Verify CDB - Unrecovered Read Error; and
f) 09-380r1 - SBC: FORMAT UNIT command fix.

Added a clause describing state diagram notation into clause 3, Definitions, symbols, acronyms, keywords, and conventions.

Minor editorial corrections were made, several based on email input, which are indicated by change bars.

Other editorial and formatting changes were made that may not be indicated by change bars.

## 24 Revision 22 (29 March 2010)

Incorporated the following proposals:

a) 09-100r5 - SBC-3: Atomic COMPARE AND WRITE command;
b) 09-241r6 - SBC-3, SPC-4, SAM-5: SCSI Referrals;
c) 09-272r6 - SBC-3: Thin Provisioning: Anchored;
d) 09-357r4 - SPC-4, SBC-3: Relationship between power conditions and background tasks;
e) 10-005r1 - SPC-4/SBC-3: Add Disable PI Check on xxPROTECT=0 feature; and
f) 10-044r2 - SBC-3: More Thin Provisioning clarifications.

Modified the format of the UNMAP block descriptor table based on discussion with Fred Knight.

Other minor editorial and formatting changes were made that are not indicated by change bars.

## 25 Revision 23 (30 July 2010)

Incorporated the following proposals:

a) 10-080r2 - SBC-3: TP Threshold notification to Single Initiator vs. Multi-Initiator
b) 10-133r4 - SPC-4/SBC-3: Pollable Device State model;
c) 10-137r0 - SBC-3: VRPROTECT with BYTCHK set to one — byte-by-byte comparison correction;
d) 10-148r1 - SPC-4/SBC-3: More about power conditions and background tasks;
e) 10-161r1 - SPC-4/SBC-3/SSC-4/ADC-3/SAT-3: VPD page generic format;
f) 10-179r1 - SPC-4/SBC-3: Remove implicit head of queue rule that is in SAM-5; and
g) 10-210r3 - SPC-4/SBC-3: Tweaks to power state machines found during SPL LB Resolution.

Made other minor editorial and formatting changes not indicated by change bars.

## 26 Revision 24 (05 August 2010)

Incorporated the following proposal:

a) 10-115r2 - SPC-4/SBC-3: Power condition transition

The editor erred when he included 10-210r3 in revision 23 of this draft standard, as he had assumed that all of the material from that proposal had been included in 10-115r2. However, this was not the case. So, the relevant portions of 10-115r2 that were not duplicated in 10-210r3 are now included in this draft. Because of the interactions between these two proposals, no change bars were removed from the previous revision of this draft. The indicated changes in this revision are from both revision 23 and revision 24 of this draft standard.

Made other minor editorial and formatting changes not indicated by change bars.

## 27 Revision 25 (27 October 2010)

Incorporated the following proposals:

a) 09-262r8 - SBC-3: Bitmaps on Disk - Detailed Changes;
b) 10-162r2 - SPC-4/SBC-3: Application Tag mode page;
c) 10-233r6 - SBC-3: Logical Block Provisioning Management;
d) 10-269r1 - SBC-3: Get LBA Status parameter data length fix; and
e) 10-280r1 - SPC-4/SBC-3: Reject Write Without Protection Information.

Made minor editorial corrections, several based on email input, which are indicated by change bars.

Made other minor editorial and formatting changes, which are not indicated by change bars.

## 28 Revision 26 (21 January 2011)

Incorporated the following proposals:

a) 10-084r3 - SBC-3: Limits for WRITE SAME unmapping;
b) 10-206r4 - SPC-3: Informational Exceptions Control mode page clean-up (based on 11-026r1)
c) 10-319r1 - SBC-3: Flush cache when WCE set to zero;
d) 10-334r3 - SBC-3: Adding ANCHOR/UNMAP bits to WRITE SAME (10);
e) 11-006r1 - SBC-3: Add unmap rule for MODE SELECT capacity change;
f) 11-010r0 - SBC-3/SAT-3/SPC-4: Obsolete READ CAPACITY command PMI bit;
g) 11-019r1 - SBC-3 SPC-4 - LBP cleanup;
h) 11-060r0 - SPC-4/SBC-3: Remove SAS-2 references;
i) 11-074r0 - SBC-3: Log page clean-up.

Made minor editorial corrections, several based on email input, which are indicated by change bars.

Made other minor editorial and formatting changes, which are not indicated by change bars.

## 29 Revision 27 (01 June 2011)

Incorporated the following proposals:

a) 11-011r4 - SBC-3/SPC-4: Add Sanitize command;
b) 11-024r1 - SBC-3: LBP Provisioning types;
c) 11-031r2 - SBC-3: LBP Threshold parameter adjustment;
d) 11-037r2 - SBC3: Add an LBPRZ bit in LBP VPD Page;
e) 11-038r1 - SBC-3: Reject write without protection Information;
f) 11-098r0 - SBC-3: Error in table 32;
g) 11-192r0 - SBC-3: Mode page off by one errors; and
h) 11-216r1 - SPC-4/SBC-3: Error conditions during resetting of log parameters.

Made editorial corrections, several based on email input, which are indicated by change bars, including the following:

a) changed the format of several paragraph styles to justified with hyphenation;
b) changed the format of the definitions in clause 3.1 to be more like the ISO style;
c) deleted footnote cross reference (b) from the ACTIVE row in the POWER CONDITION and POWER CONDITION MODIFIER field in the START STOP UNIT command description, because that cross reference does not apply to the ACTIVE power condition; and
d) added the PROVISIONING TYPE field to the Logical Block Provisioning VPD page. The description had been added based on a previous proposal, but the name of the field was not added to the table at that time.

Made other minor editorial and formatting changes, which are not indicated by change bars.

## 30 Revision 28 (30 September 2011)

Converted all files from FrameMaker 8 to FrameMaker 10, adjusting many formats and styles.

Made many changes to make this standard be consistent with the latest revision of the Style Guide (i.e., 10-316r6). Changes in text are indicated by change bars.

To match the request-response model parameter names in SPC-4, the capitalization of all instances of data-in buffer and data-out buffer were corrected to Data-In Buffer and Data-Out Buffer, respectively. These changes were not marked with change bars.

Incorporated the following proposals:

a) 10-360r5 - SPC-4 SBC-3: Make microcode activation events optional [Houlder];
b) 11-096r3 - SBC-3: Storage efficiency resource reporting [Knight];
c) 11-326r2 - SBC-3: GET LBA STATUS Clarifications [Knight & Black];
d) 11-327r2 - SAT-3 SBC-3 SPC-4: SAT-specific code cleanups [Elliott]; and
e) 11-350r1 - SBC-3: Clarify sanitize interaction with logical block provisioning [Black].

Made editorial corrections, several based on email input, which are indicated by change bars.

Made other minor editorial and formatting changes, which are not indicated by change bars.

## 31 Revision 29 (04 October 2011)

Accepted all indicated track text edits and cleared all change bars from the previous revision.

Incorporated proposal 11-350r1 - SBC-3: Clarify sanitize interaction with logical block provisioning [Black], which was incorrectly indicated as being included in revision 28.

Made editorial corrections, several based on email input, which are indicated by change bars.

Made other minor editorial and formatting changes, which are not indicated by change bars.

## 32 Revision 30 (13 February 2012)

Accepted all indicated text edit tracking and cleared all change bars from the previous revision.

Incorporated the following proposals marked with text edit tracking and change bars, where practicable:

a) 11-062r6 - SPC-4: Power Consumption Limit settings for SPC-4 [Geldman];
b) 11-376r3 - SBC-3: Add extension to defect descriptors [Houlder];
c) 11-377r4 - SBC-3: Add descriptor index field to READ DEFECT DATA command [Houlder];
d) 11-384r3 - SBC-3: Add 'Allow Unrestricted Sanitize Exit' to Sanitize [Geldman];
e) 11-413r4 - SBC-3: POPULATE TOKEN, select token type [Ballard];
f) 11-489r2 - SBC-3/SPC-4: Third Party Copy Commands for SBC-3 [Knight, et. al.];
g) 12-012r1 - SBC-3: XCOPYv2: Representation of Data tokens [Black];
h) 12-019r1 - SPC-4 SBC-3 SAT-3: Obsolete READ (6) and WRITE (6) [Elliott];
i) 12-036r1 - SBC-3: Zero-detect unmapping for logical block provisioning [Black/Knight]; and
j) 12-052r1 - SBC-3: Adding a VS field to the OVERWRITE service action parameter list [Evans].

Made editorial corrections, several based on email input, which are indicated by text edit tracking and/or change bars.

Made other minor editorial and formatting changes, which are not indicated by change bars.

## 33 Revision 31 (25 May 2012)

Accepted all indicated text edit tracking and cleared all change bars from the previous revision.

Incorporated the following proposals marked with text edit tracking and change bars, where practicable:

a) 11-298r7 - SPC-4, SBC-3: Add rebuild assist feature [Houlder];
b) 12-032r2 - SPC-4, SBC-3: Advise migrating to larger CDBs [Elliott];
c) 12-036r1 - SBC-3: Zero-detect unmapping for logical block provisioning [Black/Knight] [this proposal was not included in r30 as was noted above];
d) 12-142r1 - SBC-3: New definition for 00b in the TEST field [Evans];
e) 12-098r1 - SBC-3: Third Party Copy corrections [Knight];
f) 12-164r1 - SPC-4, SBC-3, SPL-2: Power Condition mode page field names [Elliott];
g) 12-166r1 - SBC-3, SPC-4: Third Party Copy Additions [Knight];
h) 12-186r1 - SBC-3 task set clarifications [Elliott]; and
i) 12-191r1 - SBC-3 Advise migrating from XDWRITE and XDREAD to XDWRITEREAD [Elliott] [the final revision of this proposal that was accepted at the CAP working group made the XDREAD and XDWRITE commands obsolete].

Added the convention that, if there is more than one CDB length for a command name, then the name is used without a length to define that the description is for all of commands with that name. Many instances were changed without change indication to reflect this new convention.

Made editorial corrections, several based on email input, which are indicated by text edit tracking and/or change bars.

Made other minor editorial and formatting changes, which are not indicated by change bars.

## 34 Revision 32 (06 September 2012)

Accepted all indicated text edit tracking and cleared all change bars from the previous revision.

Incorporated the following proposals marked with text edit tracking and change bars, where practicable:

a) 12-272r0 - SBC-3 SAT-3 Obsolete WRITE SAME command LBDATA and PBDATA bits [Elliott];
b) 12-273r0 - SBC-3 SAT-3 Obsolete FORMAT UNIT command IP MODIFIER field [Elliott]; and
c) 12-275r0 - SBC-3 Obsolete XOR Control mode page [Elliott].

Made editorial corrections, several based on email input, which are indicated by text edit tracking and/or change bars.

Made other minor editorial and formatting changes, which are not indicated by change bars.

## 35 Revision 33 (31 October 2012)

Accepted all indicated text edit tracking and cleared all change bars from the previous revision.

Incorporated the following proposals marked with text edit tracking and change bars, where practicable:

a) 12-104r6 - SPC-5 SBC-4 Add progress indication description to SYNCHRONIZE CACHE [Houlder];
b) 12-225r4 - SBC-3 Clarify behavior when unable to unmap [Ballard];
c) 12-274r2 - SBC-3 Migrate from the FORMAT UNIT command SI bit to SANITIZE [Elliott]; and
d) 12-320r2 - SBC-3 LBA out of range and transfer length cleanup [Elliott].

Made editorial corrections, several based on email input, which are indicated by text edit tracking and/or change bars.

Made other minor editorial and formatting changes, which are not indicated by change bars.

## 36 Revision 34 (30 November 2012)

Accepted all indicated text edit tracking and cleared all change bars from the previous revision.

Incorporated the following proposals marked with text edit tracking and change bars, where practicable:

a) 12-139r5 - SBC-3: Definitions for read, write, verify, and unmap operations [Evans];
b) 12-246r4 - SBC-3: VERIFY SAME command [Knight];
c) 12-282r4 - SBC-3: Allow verify operations on unmapped LBAs [Elliott];
d) 12-283r3 - SBC-3: REASSIGN BLOCKS simplifications for logical block provisioning [Elliott];
e) 12-320r5 - SBC-3: Sanitize operation definition clarification and expansion [Evans];
f) 12-368r2 - SBC-3: GET LBA STATUS and related clarifications [Elliott];
g) 12-398r0 - SBC-3: Remove attached media changer remnants [Elliott];
h) 12-403r1 - SBC-3: Clarifications to POPULATE TOKEN and UNMAP [Knight];
i) 12-406r0 - SBC-3: Fix for SPC-4 LB comment HP-255 [Knight];
j) 12-420r1 - SBC-3: Clearing pseudo unrecovered errors [Elliott];
k) 12-442r1 - SPC-4, SBC-3: Add Media Type in Block Device Characteristics VPD page [Chen, Geldman, and Landsman]; and
l) 13-001r0 (nee 11-030r4) - SBC-3: LBP Threshold example Annex [Knight].

Made editorial corrections, several based on email input, which are indicated by text edit tracking and/or change bars.

Made other minor editorial and formatting changes, which are not indicated by change bars.

## 37 Revision 34a (30 November 2012)

Accepted all indicated text edit tracking and cleared all change bars from the previous revision.

Made minor editorial corrections and formatting changes, which are not indicated by change bars.

## 38 Revision 34b (07 December 2012)

Made minor editorial corrections and formatting changes, most of which are indicated by change bars.

## 39 Revision 35 (07 December 2012)

Cleared all change bars from the previous revision.

## Contents

## Tables

Page

## Figures

Page

## Foreword

This foreword is not part of American National Standard BSR INCITS 514.

This purpose of this standard is to define the model and command set extensions to be used in conjunction with the SCSI Primary Command Set standard – 4 (SPC-4) to facilitate operation of SCSI direct-access block devices (e.g., hard disk drives).

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

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

Karen Higginbottom, Chair

David Michael, Vice-Chair

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

John B. Lohmeyer, Chair

Mark S. Evans, Vice-Chair

Ralph O. Weber, Secretary

## Introduction

The standard is organized as follows:

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

Informative Annex A (Numeric order codes) summarizes service action assignments for variable-length commands and commands using the SERVICE ACTION IN and SERVICE ACTION OUT operation codes.
Informative Annex B (XOR command examples) provides examples of XOR command usage.
Informative Annex C (CRC example in C) provides example C code for the protection information CRC.
Informative Annex D (Sense information for locked or encrypted SCSI target devices)
Informative Annex E (Optimizing block access characteristics)
Informative Annex F (Logical block provisioning reporting)

## SCSI standards family

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

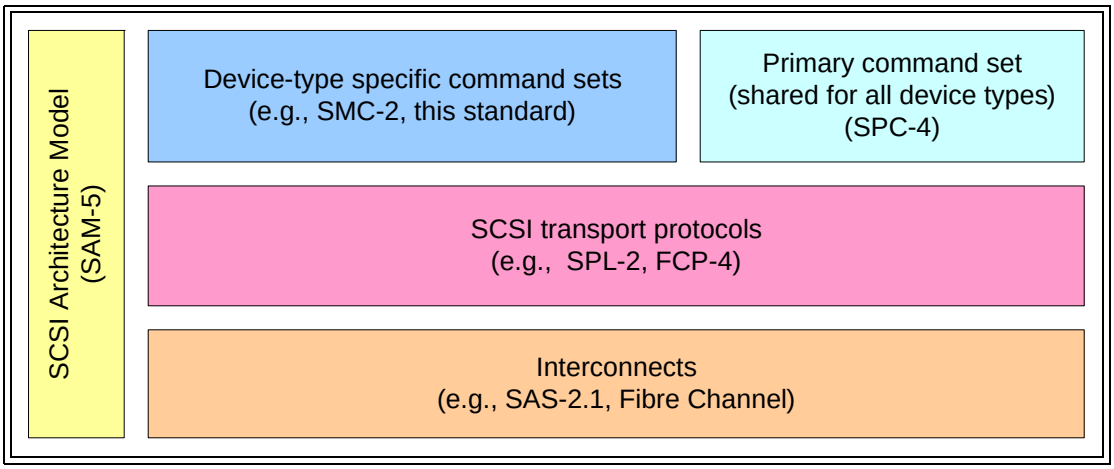



**Figure 0 — SCSI document relationships**

Figure 0 is intended to show the general relationship of the documents to one another and is not intended to imply a relationship such as a hierarchy, protocol stack, or system architecture.

The set of SCSI standards specifies the interfaces, functions, and operations necessary to ensure interoperability between conforming SCSI implementations. This standard is a functional description. Conforming implementations may employ any design technique that does not violate interoperability. See SAM-5 for more information about the relationships between the SCSI standards.

This standard makes obsolete the following concepts from previous standards:

a) linked commands;
b) the partial medium indicator (PMI) bit and the LOGICAL BLOCK ADDRESS field in the READ CAPACITY commands;
c) the READ (6) command and the WRITE (6) command;
d) the XDREAD commands and the XDWRITE commands;
e) the LBDATA bit and the PBDATA bit in the WRITE SAME commands;
f) the IP MODIFIER (initialization pattern modifier) field in the initialization pattern descriptor in the FORMAT UNIT command; and
g) the XOR Control mode page.

**American National Standard**
**for Information Technology -**

# SCSI Block Commands – 3 (SBC-3)

## 1 Scope

This standard defines the command set extensions to facilitate operation of SCSI direct-access block devices. The clauses in this standard, implemented in conjunction with the applicable clauses of SPC-4, specify the standard command set for SCSI direct-access block devices.

The objective of this standard is to:

a) permit an application client to communicate over a SCSI service delivery subsystem with a logical unit that declares itself to be a direct-access block device in the PERIPHERAL DEVICE TYPE field of the standard INQUIRY data (see SPC-4); and
b) define commands unique to the direct-access block device type.

## 2 Normative references

### 2.1 Normative references overview

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

Copies of the following documents may be obtained from ANSI:

a) approved ANSI standards;
b) approved and draft international and regional standards (e.g., ISO, IEC, CEN/CENELEC, ITU-T); and
c) approved and draft foreign standards (e.g., BSI, JIS, and DIN).

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

Additional availability contact information is provided below as needed.

### 2.2 Approved references

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

ISO/IEC 14776-342, *SCSI-3 Controller Commands - 2 (SCC-2)* (ANSI INCITS 318-1998)
ISO/IEC 14776-414, *SCSI Architecture Model - 4 (SAM-4)* (ANSI INCITS 447-2008)
INCITS 468-2010, Information Technology - Multi-Media Commands - 6 (MMC-6)
INCITS 468-2010/AM 1 Draft, Information technology - MultiMedia Command Set - 6/AMENDMENT 1 (MMC-6/AM 1)

### 2.3 References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the documents, or regarding availability, contact the relevant standards body as indicated.

ISO/IEC 14776-415, *SCSI Architecture Model - 5 (SAM-5)* (T10/2104-D)
ISO/IEC 14776-454, *SCSI Primary Commands - 4 (SPC-4)* (T10/1731-D)
ISO/IEC 14776-372, *SCSI Enclosure Services - 2 (SES-2)* (T10/1559-D)
ISO/IEC 14776-923, *SCSI / ATA Translation - 3* (SAT-3) (T10/2126-D)

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

## 3.1 Definitions, symbols, abbreviations, keywords, and conventions Introduction

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

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

| Topic | Reference |
|---|---|
| Definitions | 3.2 |
| Symbols | 3.3 |
| Abbreviations | 3.4 |
| Keywords | 3.5 |
| Editorial conventions | 3.6 |
| Numeric conventions | 3.7 |
| State machine conventions | 3.8 |

## 3.2 Definitions

**3.2.1 additional sense code (see SPC-4)**
combination of the ADDITIONAL SENSE CODE field and the ADDITIONAL SENSE CODE QUALIFIER field in the sense data

**3.2.2 AND**
Boolean arithmetic function (see 3.2.10) on two binary input values that results in an output value of one if both of the input values are one, or zero if either of the input values is zero

**3.2.3 AND operation**
performance of an AND (see 3.2.2) bitwise on two multiple-bit input values both having the same number of bits (e.g., on the current content of a logical block and the content contained in the Data-Out Buffer having the same number of bytes)

**3.2.4 anchored**
logical block provisioning relationship state of an LBA (see 4.7.1) for which the logical unit has reserved physical capacity (see 4.7.4.5)

**3.2.5 application client (see SPC-4)**
object that is the source of SCSI commands

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

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

**3.2.8 background function**
either a background scan operation (see 4.24) or a device specific background function (see 3.2.23)

**3.2.9 bitmap buffer**
temporary buffer within a device server (e.g., for one or more bytes of the result of an AND operation (see 3.2.3) or an OR operation (see 3.2.53))

**3.2.10 Boolean arithmetic function**
function that produces an output from one or more inputs according to the rules of Boolean algebra (see 3.2.2, 3.2.28, and 3.2.52)

**3.2.11 byte**
sequence of eight contiguous bits considered as a unit

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

**3.2.13 check data**
information contained within a redundancy group (see 3.2.73) that may allow lost or destroyed XOR-protected data (see 3.2.100) to be recreated

**3.2.14 command (see SAM-5)**
request describing a unit of work to be performed by a device server

**3.2.15 command descriptor block (CDB) (see SPC-4)**
structure used to communicate commands from an application client to a device server

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

**3.2.17 cyclic redundancy check (CRC)**
error checking mechanism that checks data integrity by computing a polynomial algorithm based checksum (see 4.22.4)

**3.2.18 Data-In Buffer (see SAM-5 and SPC-4)**
buffer identified by the application client to receive data from the device server during the processing of a command

**3.2.19 Data-Out Buffer (see SAM-5 and SPC-4)**
buffer identified by the application client to supply data that is sent from the application client to the device server during the processing of a command

**3.2.20 deallocated**
logical block provisioning state of an LBA (see 4.7.1) for which the logical unit has not reserved physical capacity (see 4.7.4.4)

**3.2.21 defect**
location of one or more physical blocks that the device server has determined contains a medium defect or for which an application client has requested a reassign operation

**3.2.22 device server (see SAM-5)**
object within a logical unit (see 3.2.44) that processes SCSI commands and some task management functions

**3.2.23 device specific background functions (see SPC-4)**
SCSI target device specific functions that a device may perform when there are no other application client-initiated operations in progress

**3.2.24 device type**
type of device (or device model) implemented by the device server as indicated by the PERIPHERAL DEVICE TYPE field of the standard INQUIRY data (see SPC-4)

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

**3.2.26 domain (see SAM-5)**
I/O system consisting of a set of SCSI devices that interact with one another by means of a service delivery subsystem

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

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

**3.2.29 extent**
fixed set of logical blocks occupying contiguous LBAs on a single logical unit

**3.2.30 field**
group of one or more contiguous bits, a part of a larger structure such as a CDB (see 3.2.15) or sense data (see SPC-4)

**3.2.31 format corrupt**
vendor specific condition in which the device server may not be able to perform read medium operations, unmap operations, verify medium operations, or write medium operations (see 4.10)

**3.2.32 format operation (see 4.10)**
process by which a device server initializes the medium in a logical unit

**3.2.33 fully provisioned logical unit (see 4.7.2)**
logical unit that stores logical block data for every LBA

**3.2.34 grown defect list (GLIST)** (see 4.13)
list of physical blocks that the device server has detected as containing medium defects or that the application client has specified as containing medium defects

**3.2.35 hard reset**
condition resulting from the events defined by SAM-5 in which the SCSI device performs the hard reset operations described in SAM-5, this standard, and other applicable command standards (see table 31 in 5.1)

**3.2.36 I_T nexus (see SAM-5)**
relationship between a SCSI initiator port and a SCSI target port

**3.2.37 I_T nexus loss**
condition resulting from the events defined by SAM-5 in which the SCSI device performs the I_T nexus loss operations described in SAM-5, this standard, and other applicable command standards (see table 31 in 5.1)

**3.2.38 LBA mapping resource**
resource (e.g., a physical block or a data structure associated with tracking resource usage) used by a logical unit that supports logical block provisioning management

**3.2.39 logical block**
set of data bytes accessed and referenced as a unit (see 4.5)

**3.2.40 logical block access command (see 4.2.2)**
command that requests access to one or more logical blocks that may require access to the medium

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

**3.2.42 logical block data**
user data and protection information, if any

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

**3.2.44 logical unit (see SAM-5)**
externally addressable entity within a SCSI target device (see 3.2.76) that implements a SCSI device model and contains a device server

**3.2.45 logical unit reset**
condition resulting from the events defined by SAM-5 in which the logical unit performs the logical unit reset operations described in SAM-5, this standard, and other applicable command standards (see table 31 in 5.1)

**3.2.46 mapped**
logical block provisioning state of an LBA (see 4.7.1) for which the logical unit has assigned physical capacity to specific logical block(s) (see 4.7.4.3)

**3.2.47 media**
plural of medium

**3.2.48 medium**
material that is not cache on which data is stored (e.g., a magnetic disk)

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

**3.2.50 non-volatile cache**
cache that retains data through power cycles

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

**3.2.52 OR**
Boolean arithmetic function (see 3.2.10) on two binary input values that results in an output value of one if either of the input values are one, or zero if both of the input values are zero

**3.2.53 OR operation**
performance of an OR (see 3.2.52) bitwise on two multiple-bit input values both having the same number of bits (e.g., on the current content of a logical block and the content contained in the Data-Out Buffer having the same number of bytes)

**3.2.54 point in time ROD token (see SPC-4)**
ROD token with a ROD type that is a point in time copy ROD

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

**3.2.56 physical block length**
number of bytes of user data in a physical block (see 4.6)

**3.2.57 physical element**
component that provides non-volatile storage for an associated group of logical blocks (see 4.20)

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

**3.2.59 power on**
condition resulting from the events defined by SAM-5 in which the SCSI device performs the power on operations described in SAM-5, this standard, and other applicable command standards (see table 31 in 5.1)

**3.2.60 primary defect list (PLIST) (see 4.13)**
list of physical blocks containing medium defects that are considered permanent

**3.2.61 protection information**
fields appended to each logical block or added at specified intervals within a logical block that contain a cyclic redundancy check (CRC), an application tag, and a reference tag (see 4.22)

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

**3.2.63 pseudo read data**
data that is transferred by a device server based on the setting of the RC bit or the TB bit in the Read-Write Error mode page (see 6.5.8) in order to maintain a continuous flow of data or to transfer the amount of data requested for a command even though an unrecovered read error occurred while the device server was processing the command, which results in the data being indeterminate (e.g., data already in a buffer or any other vendor specific data)

**3.2.64 pseudo unrecovered error**
simulated error (e.g., created by a WRITE LONG command (see 5.41 and 5.42)) for which a device server reports that it is unable to read or write a logical block, regardless of whether the data on the medium is valid, recoverable, or unrecoverable

**3.2.65 pseudo unrecovered error with correction enabled**
pseudo unrecovered error (see 3.2.64) for which a device server performs the maximum error recovery as specified in the Read-Write Error Recovery mode page (see 6.5.8) (see 4.18.2)

**3.2.66 pseudo unrecovered error with correction disabled**
pseudo unrecovered error (see 3.2.64) for which a device server performs no error recovery (see 4.18.2)

**3.2.67 read command (see 4.2.2)**
command that requests access to logical blocks that may require read medium operations

**3.2.68 read medium operation**
process by which a device server reads one or more logical blocks from the medium using the parameters specified in the Read-Write Error Recovery mode page (see 6.5.8)

**3.2.69 reassign**
perform a reassign operation

**3.2.70 reassign operation**
operation during which the device server changes the assignment of an LBA from one physical block to another physical block

**3.2.71 recovered error**
error for which a device server is able to read or write a logical block within the recovery limits specified in the Read-Write Error Recovery mode page (see 6.5.8) and the Verify Error Recovery mode page (see 6.5.9)

**3.2.72 recovered read error**
recovered error that occurs during a read medium operation

**3.2.73 redundancy group**
grouping of XOR-protected data (see 3.2.100) and associated check data (see 3.2.13) into a single type of data redundancy (see SCC-2). This standard only supports the XOR (see 3.2.28) type of redundancy

**3.2.74 resource provisioned logical unit (see 4.7.3.2)**
logical unit that may or may not store user data and protection information for every LBA and that provides enough LBA mapping resources (see 3.2.38) to map every LBA

**3.2.75 sanitize operation (see 4.11)**
process by which a device server alters information on a logical unit such that recovery of logical block data from the cache and the medium is not possible

**3.2.76 SCSI target device (see SAM-5)**
SCSI device containing logical units and SCSI target ports that receives device service and task management requests for processing and sends device service and task management responses to SCSI initiator devices

**3.2.77 sense data (see SPC-4)**
data describing an error or exceptional condition that a device server delivers to an application client

**3.2.78 sense key (see SAM-5)**
contents of the SENSE KEY field in the sense data

**3.2.79 status (see SAM-5)**
one byte of response information sent from a device server to an application client upon completion of each command

**3.2.80 stopped power condition**
power condition in which a device server terminates TEST UNIT READY commands and medium access commands (see 4.21.1).

**3.2.81 thin provisioned logical unit (see 4.7.3.3)**
logical unit that may or may not store user data and protection information for every LBA and that may or may not provide enough LBA mapping resources (see 3.2.38) to map every LBA

**3.2.82 threshold set (see 4.7.3.8)**
set of two or more logical blocks used for tracking logical block provisioning thresholds

**3.2.83 threshold set size**
number of LBAs in a threshold set

**3.2.84 token (see SPC-4)**
representation of a collection of data

**3.2.85 unit attention condition (see SAM-5)**
state that a logical unit (see 3.2.44) maintains while the logical unit has asynchronous status information to report to the initiator ports associated with one or more I_T nexuses (see 3.2.36)

**3.2.86 unmap operation (see 4.7.3.4)**
process by which a device server either deallocates or anchors a single LBA

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

**3.2.88 unrecovered error**
error for which a device server is unable to read or write a logical block within the recovery limits specified in the Read-Write Error Recovery mode page (see 6.5.8) and/or the Verify Error Recovery mode page (see 6.5.9)

**3.2.89 unrecovered read error**
unrecovered error that occurs during a read medium operation

**3.2.90 user data**
data contained in logical blocks that is neither protection information nor other information that may not be accessible to the application client

**3.2.91 user data segment (see 4.28)**
contiguous sequence of logical blocks

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

**3.2.93 verify command (see 4.2.2)**
command that requests verify medium operations that may require verify medium operations

**3.2.94 verify medium operation**
kind of read medium operation in which the device server uses the parameters specified in the Verify Error Recovery mode page (see 6.5.9) rather than the parameters specified in the Read-Write Error Recovery mode page (see 6.5.8)

**3.2.95 volatile cache**
cache that does not retain data through power cycles (e.g., a silicon memory device that does not retain data written to it if device power is lost)

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

**3.2.97 write command (see 4.2.2)**
command that requests access to logical blocks that may require write medium operations

**3.2.98 write medium operation**
process by which a device server writes one or more logical blocks to the medium using the parameters specified in the Read-Write Error Recovery mode page (see 6.5.8) and which may cause allocation of LBA mapping resources (see 4.7.1)

**3.2.99 XOR operation**
an XOR bitwise on two identical-sized multiple-bit input values (e.g., the current value of a logical block and the new value for that logical block)

**3.2.100 XOR-protected data**
logical blocks, including logical block data that are part of a redundancy group

## 3.3 Symbols

Symbols used in this standard include:

| Symbol | Meaning |
|---|---|
| + | plus |
| – | minus |
| × | multiplied by |
| ÷ | divided by |
| = | equals |
| < | less than |
| > | greater than |

## 3.4 Abbreviations

Abbreviations used in this standard include:

| Abbreviation | Meaning |
|---|---|
| CDB | command descriptor block (see 3.2.15) |
| CRC | cyclic redundancy check (see 3.2.17) |
| ECC | error correcting code (see 3.2.27) |
| GLIST | grown defect list (see 3.2.34) |
| I/O | input/output |
| LBA | logical block address (see 3.2.41) |
| LBP | logical block provisioning (see 4.7.4) |
| LSB | least significant bit |
| LUN | logical unit number |
| M | implementation is mandatory |
| MMC-6 | SCSI Multimedia Commands - 6 standard |
| MSB | most significant bit |
| O | implementation is optional |
| PLIST | primary defect list (see 3.2.60) |
| n/a | not applicable |
| ROD | representation of data (see SPC-4) |
| rpm | revolutions per minute |
| SAM-5 | SCSI Architecture Model - 5 standard |
| SCSI | Small Computer System Interface family of standards |
| SCC-2 | SCSI-3 Controller Commands - 2 standard |
| SES-2 | SCSI Enclosure Services - 2 standard |
| SPC-4 | SCSI Primary Commands - 4 standard |
| X | implementation requirements are defined in the reference |
| XOR | exclusive-or (see 3.2.28) |

## 3.5 Keywords

### 3.5.1 ignored

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

**3.5.2 invalid**
a keyword used to describe an illegal or unsupported bit, byte, word, field or code value; receipt of an invalid bit, byte, word, field or code value shall be reported as an error

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

**3.5.4 may**
a keyword that indicates flexibility of choice with no implied preference; “may” is equivalent to “may or may not”

**3.5.5 may not**
keywords that indicate flexibility of choice with no implied preference; “may not” is equivalent to “may or may not”

**3.5.6 need not**
keywords indicating a feature that is not required to be implemented; “need not” is equivalent to “is not required to”

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

**3.5.8 optional**
a keyword that describes features that are not required to be implemented by this standard. However, if any optional feature defined in this standard is implemented, then it shall be implemented as defined in this standard

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

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

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

**3.5.12 shall**
a keyword indicating a mandatory requirement; designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard

**3.5.13 should**
a keyword indicating flexibility of choice with a strongly preferred alternative; “should” is equivalent to the phrase “it is strongly recommended”

## 3.6 Editorial conventions

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## 3.7 Numeric conventions

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

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

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

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

This standard uses the following conventions for representing decimal numbers:

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

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

**Table 2 — Numbering convention examples**

| French | English | This standard |
|---|---|---|
| 0,6 | 0.6 | 0.6 |
| 3,141 592 65 | 3.14159265 | 3.141 592 65 |
| 1 000 | 1,000 | 1 000 |
| 1 323 462,95 | 1,323,462.95 | 1 323 462.95 |

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

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

## 3.8 State machine conventions

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

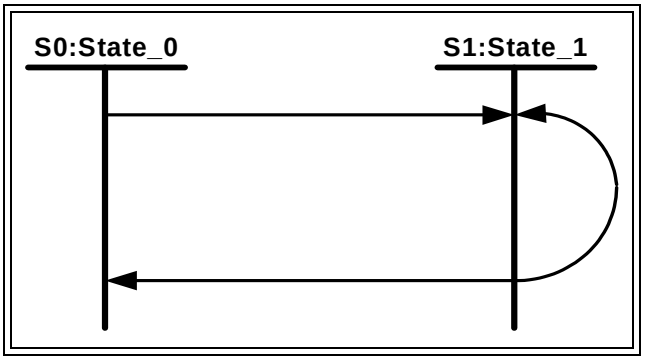



**Figure 1 — Example state machine figure**

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

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

A system specified in this manner has the following properties:

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

# 4 Direct-access block device type model

## 4.1 Direct-access block device type model introduction

Table 3 shows the topics in clause 4 and a reference to the subclause where each topic is described.

**Table 3 — Direct-access block device type mode topics**

| Topic | Reference |
|---|---|
| Direct-access block device type model overview | 4.2 |
| Media examples | 4.3 |
| Removable media | 4.4 |
| Logical blocks | 4.5 |
| Physical blocks | 4.6 |
| Logical block provisioning | 4.7 |
| Ready state | 4.9 |
| Initialization | 4.10 |
| Sanitize operations | 4.11 |
| Write protection | 4.12 |
| Medium defects | 4.13 |
| Write failures | 4.14 |
| Caches | 4.15 |
| Implicit HEAD OF QUEUE command processing | 4.16 |
| Reservations | 4.17 |
| Error reporting | 4.18 |
| Model for XOR commands | 4.19 |
| Rebuild assist mode | 4.20 |
| START STOP UNIT and power conditions | 4.21 |
| Protection information model | 4.22 |
| Grouping function | 4.23 |
| Background scan operations | 4.24 |
| Association between commands and CbCS permission bits | 4.25 |
| Deferred microcode activation | 4.26 |
| Model for uninterrupted sequences on LBA ranges | 4.27 |
| Referrals | 4.28 |
| ORWRITE commands | 4.29 |
| Block device ROD token operations | 4.30 |

## 4.2 Direct-access block device type model

### 4.2.1 Direct-access block device type model overview

SCSI devices that conform to this standard are referred to as direct-access block devices. This includes the category of logical units commonly referred to as hard disk drives and removable rigid disks.

This standard is intended to be used in conjunction with SAM-5, SPC-4, SCC-2, and SES-2.

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

Logical blocks are stored by a process that causes localized changes or transitions within a medium. The changes made to the medium to store the logical blocks may be volatile (i.e., not retained through power cycles) or non-volatile (i.e., retained through power cycles).

### 4.2.2 Logical block access command types

The following are logical block access command types:

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

See table 31 in 5.1 for a list of commands for direct access block devices, including the logical block access command type. Some commands may be more than one type of logical block access command (e.g., a COMPARE AND WRITE command is both a read command and a write command).

## 4.3 Media examples

### 4.3.1 Media examples overview

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

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

Other types of media are possible.

### 4.3.2 Rotating media

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

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

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

A rotating medium-based, direct-access block device is ready when the disks are rotating at the correct speed, and the read-write circuitry is powered and ready to access the data. A START STOP UNIT command (see 5.25) may be required to bring the logical unit to the ready state.

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

The defect management scheme of a disk device may not be discernible through this command set, though some aspects (see 4.13) may be accessible to the application client with the READ LONG commands and the WRITE LONG commands (see 5.19, 5.20, 5.41, and 5.42).

### 4.3.3 Memory media

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

A memory medium-based, direct-access block device is ready after power on, and does not require a START STOP UNIT command (see 5.25) to bring the logical unit to a ready state.

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

The defect management scheme (e.g., ECC bytes) (see 4.13) may be accessible to the application client with the READ LONG commands and the WRITE LONG commands (see 5.19, 5.20, 5.41, and 5.42).

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

## 4.4 Removable media

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

A removable medium has an attribute of being mounted or unmounted on a suitable transport mechanism in a direct-access block device. A removable medium is mounted when the direct-access block device is capable of accessing its medium (e.g., performing read medium operations and write medium operations).

A removable medium is unmounted at any other time (e.g., during loading, unloading, or storage).

An application client may check whether a removable medium is mounted by sending a TEST UNIT READY command (see SPC-4). A direct-access block device containing a removable medium may not be accessible for read medium operations, unmap operations, verify medium operations, and write medium operations until it receives a START STOP UNIT command (see 5.25).

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

If the medium in a direct-access block device is removable, and the medium is removed, then the device server shall establish a unit attention condition with the additional sense code set to the appropriate value (e.g., NOT READY TO READY CHANGE, MEDIUM MAY HAVE CHANGED).

The PREVENT ALLOW MEDIUM REMOVAL command (see 5.10) allows an application client to restrict the unmounting of the removable medium.

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

### 4.5 Logical blocks

Logical blocks are stored on the medium. Logical blocks:

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

The only commands that an application client may use to access some or all of this additional information are READ LONG commands (see 5.19) and WRITE LONG commands (see 5.41).

The number of bytes of user data contained in each logical block is the logical block length. The logical block length is greater than or equal to one byte and should be an even number of bytes. Most direct-access block devices support a logical block length of 512 bytes and some support additional logical block lengths (e.g., 520 or 4 096 bytes). The logical block length does not include the length of protection information, if any, and additional information, if any, that are contained in the logical block. The logical block length is the same for all logical blocks in the logical unit. The LOGICAL BLOCK LENGTH field in the READ CAPACITY (10) parameter data (see 5.15.2) and the READ CAPACITY (16) parameter data (see 5.16.1) indicates the logical block length. The mode parameter block descriptor (see 6.5.2) is used by an application client to change the logical block length in direct-access block devices that support changeable logical block lengths.

Each logical block is referenced by a unique LBA, which is represented as either four bytes in length or eight bytes in length. The LBAs on a logical unit are mapped, deallocated, or anchored (see 4.7). The LBAs on a logical unit shall begin with zero and shall be contiguous up to the last LBA on the logical unit. The last LBA is [n-1], where [n] is the number of logical blocks accessible by the application client. The RETURNED LOGICAL BLOCK ADDRESS field in the READ CAPACITY (10) parameter data (see 5.15.2) and the READ CAPACITY (16) parameter data (see 5.16.2) indicates the value of [n-1].

Some commands support only four-byte LOGICAL BLOCK ADDRESS fields (e.g., READ CAPACITY (10), READ (10), and WRITE (10)). If the capacity exceeds that accessible with four-byte LBAs, then the device server returns a capacity of FFFF_FFFFh in the READ CAPACITY (10) parameter data, indicating that an application client should:

a) enable descriptor format sense data (see SPC-4) in the Control mode page (see SPC-4) and in any REQUEST SENSE commands (see SPC-4) it sends; and
b) use commands with eight-byte LOGICAL BLOCK ADDRESS fields (e.g., READ CAPACITY (16), READ (16), and WRITE (16)).

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

If a command is received that references or attempts to access a logical block that exceeds the capacity of the medium (i.e., the specified LBA plus the specified length minus one is greater than the LBA indicated in the RETURNED LOGICAL BLOCK ADDRESS field in the READ CAPACITY (10) parameter data and the READ CAPACITY (16) parameter data), then the device server terminates the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The device server:

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

The location of a logical block on the medium is not required to have a relationship to the location of any other logical block. However, in a typical direct-access block device, the time to access a logical block at LBA [x+1] after accessing LBA [x] is often less than the time to access some other logical block.

### 4.6 Physical blocks

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

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

Each physical block may include additional information (e.g., an ECC), which may not be accessible to the application client.

If the device server supports the COR_DIS bit and/or the WR_UNCOR bit in a WRITE LONG command (see 5.41 and 5.42), then the device server shall have the capability of marking individual logical blocks as containing pseudo unrecovered errors with correction enabled or with correction disabled.

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

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

Figure 2 shows examples of where there are one or more physical blocks per logical block, and LBA 0 is aligned to a physical block boundary. The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field is defined in the READ CAPACITY (16) parameter data (see 5.16.2).

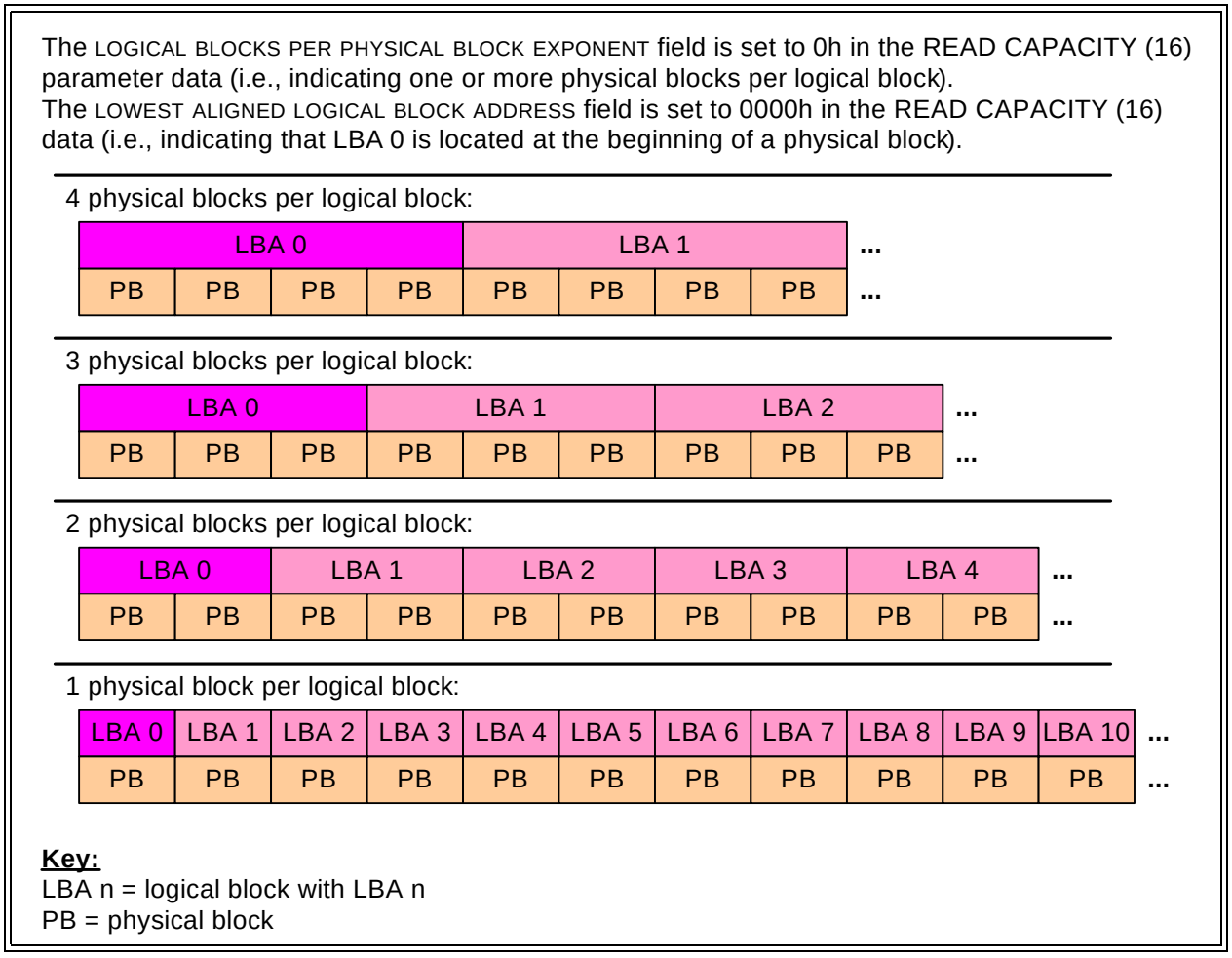



**Figure 2 — One or more physical blocks per logical block examples**

Figure 3 shows examples of where there are one or more logical blocks per physical block, and LBA 0 is aligned to a physical block boundary. The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field is defined in the READ CAPACITY (16) parameter data (see 5.16.2).

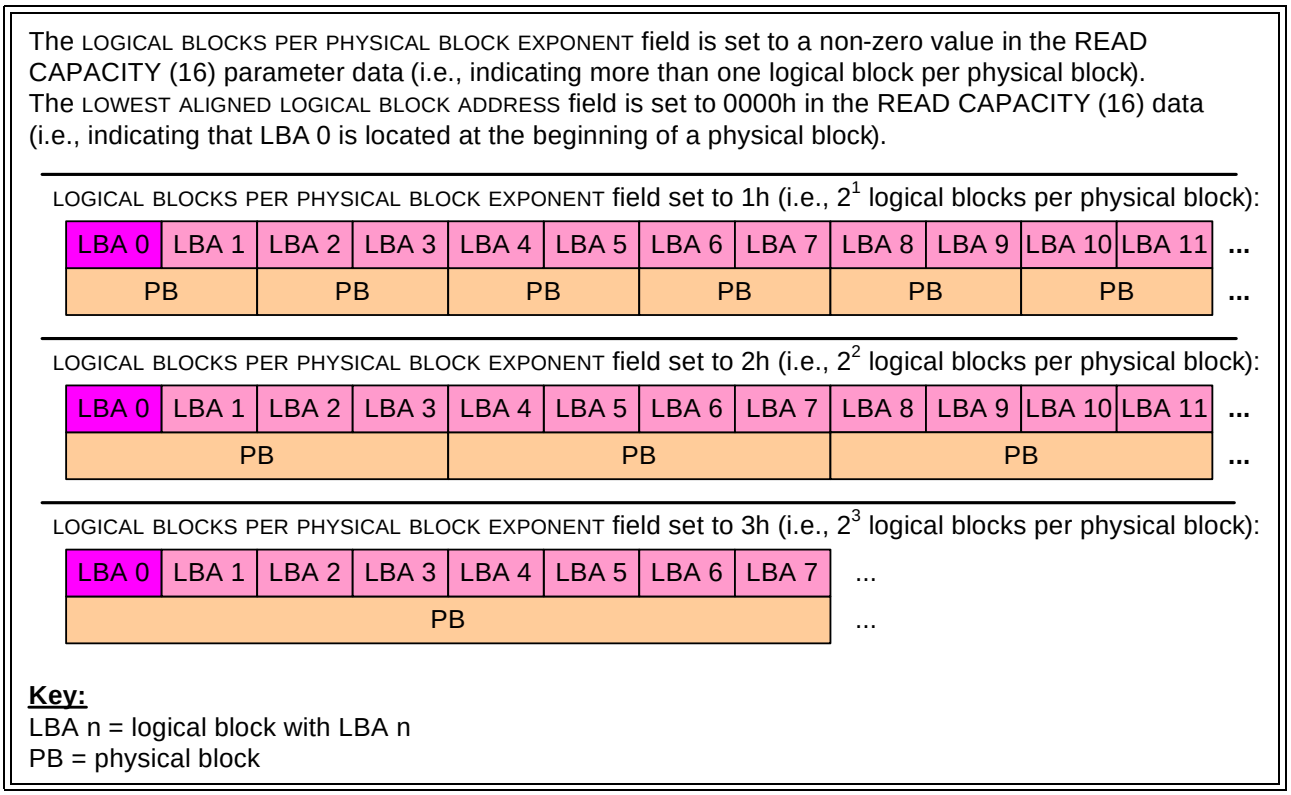



**Figure 3 — One or more logical blocks per physical block examples**

Figure 4 shows examples of where there two logical blocks per physical block, and different LBAs are aligned to physical block boundaries. The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field is defined in the READ CAPACITY (16) parameter data (see 5.16.2).

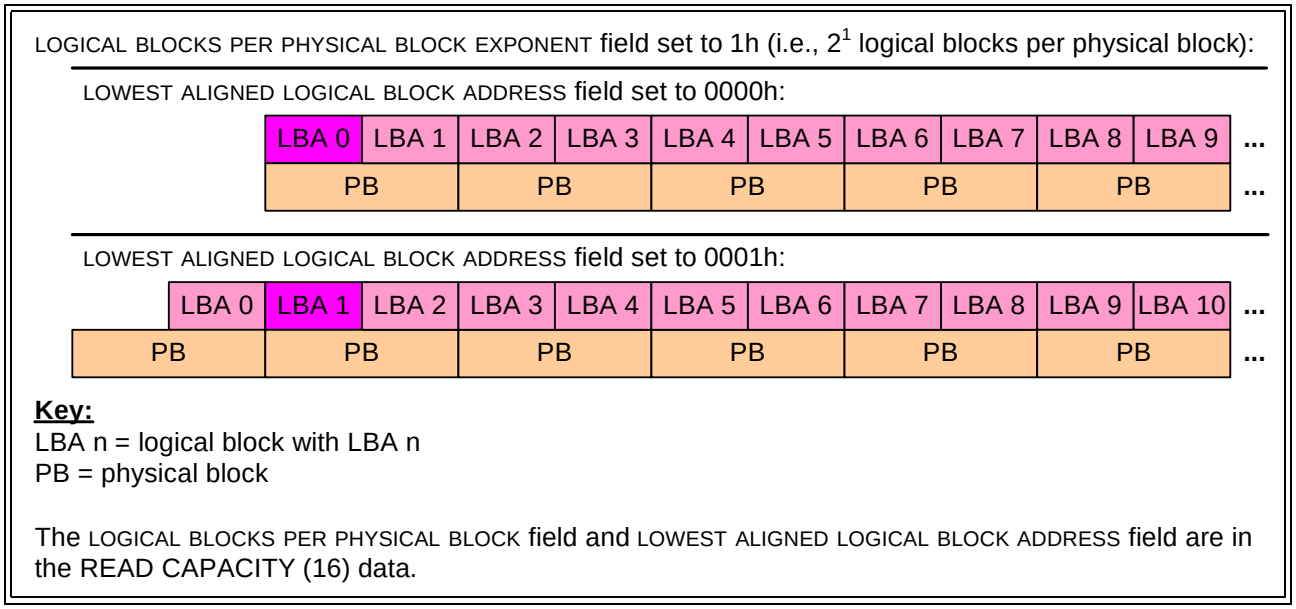



**Figure 4 — Two logical blocks per physical block alignment examples**

Figure 5 shows examples of where there four logical blocks per physical block, and different LBAs are aligned to physical block boundaries. The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field is defined in the READ CAPACITY (16) parameter data (see 5.16.2).

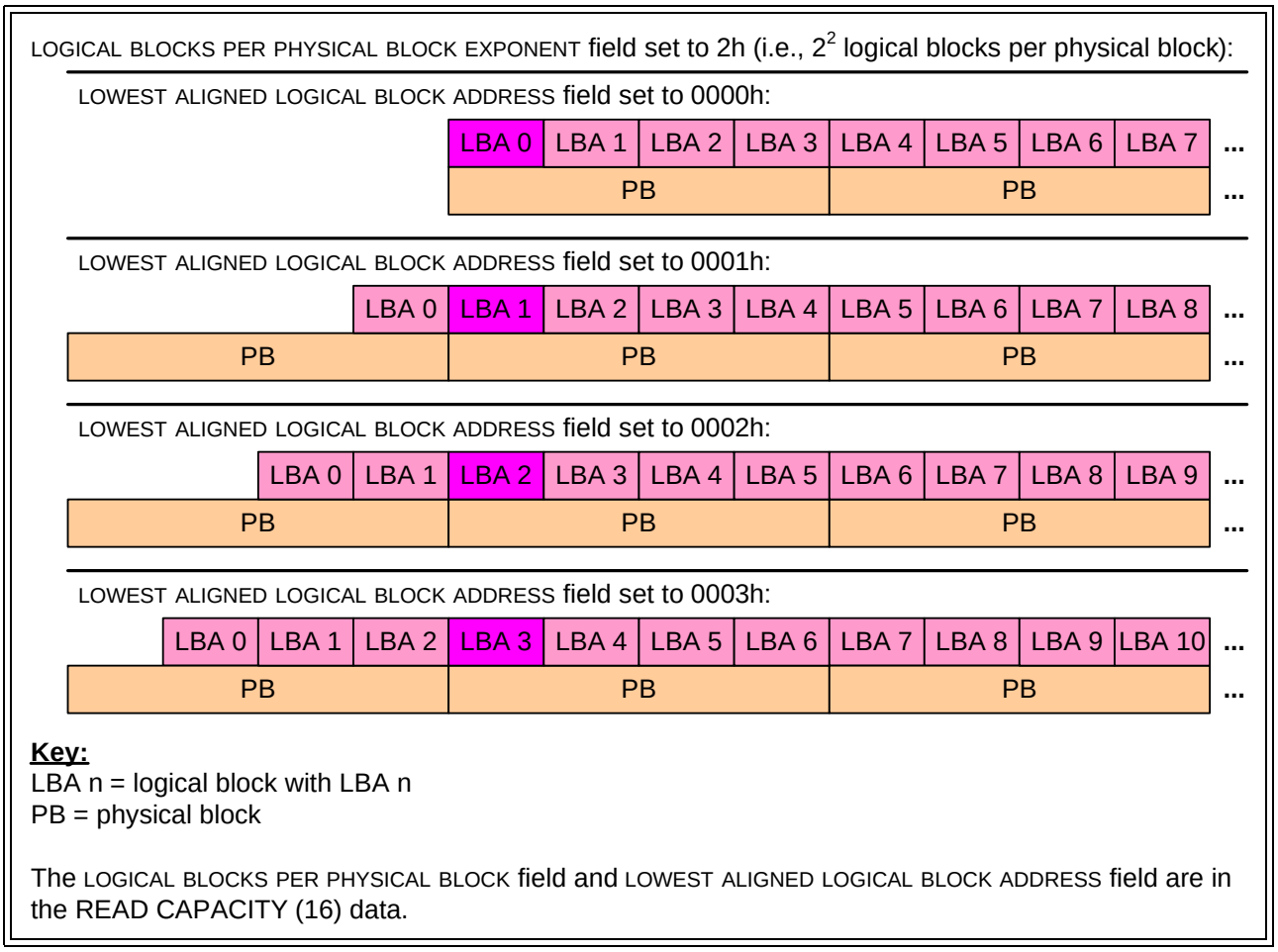



**Figure 5 — Four logical blocks per physical block alignment examples**

When there are more than one logical block per physical block, not all of the logical blocks are aligned to the physical block boundaries. When using medium access commands, application clients should:

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

## 4.7 Logical block provisioning

### 4.7.1 Logical block provisioning overview

Each LBA in a logical unit is either mapped or unmapped. For LBAs that are mapped, there is a known relationship between the LBA and one or more physical blocks that contain logical block data. For LBAs that are unmapped, the relationship between the LBA and a physical block is not defined. Figure 6 shows two examples of the relationship between mapped and unmapped LBAs and physical blocks in a logical unit. One example shows one LBA per physical block, and one example shows two LBAs per physical block. The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field is defined in the READ CAPACITY (16) parameter data (see 5.16.2).

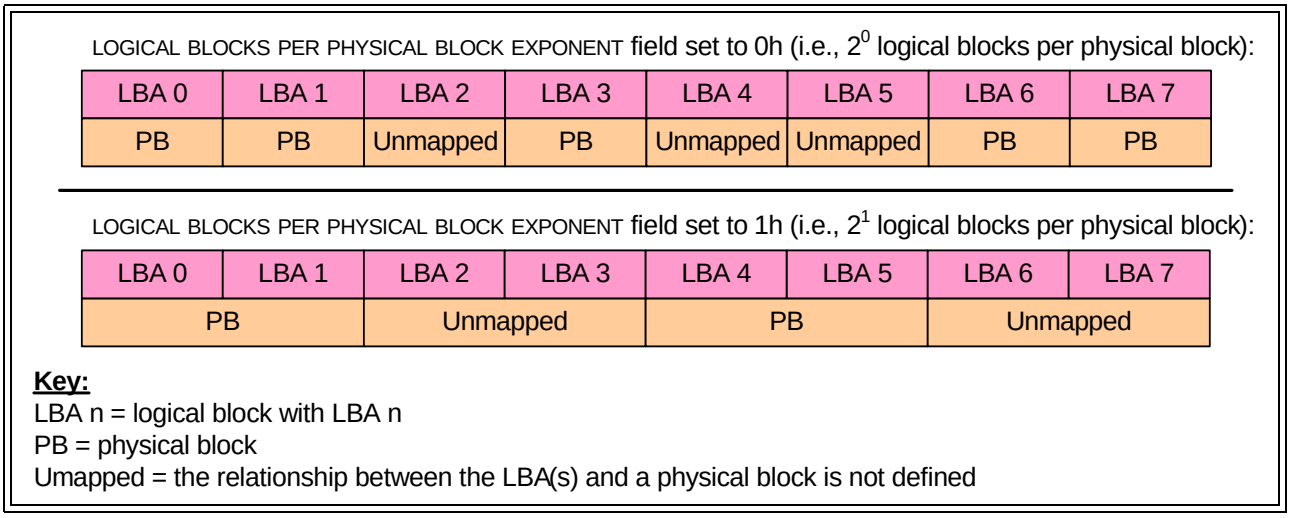



**Figure 6 — Examples of the relationship between mapped and umapped LBAs and physical blocks**

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

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

The quantity of LBA mapping resources available to a logical unit may be greater than, equal to, or less than the quantity required to store logical block data for every LBA.

### 4.7.2 Fully provisioned logical unit

Every LBA in a fully provisioned logical unit shall be mapped (i.e., stores user data and protection information, if any). A fully provisioned logical unit shall provide enough LBA mapping resources to contain all logical blocks for the logical unit's capacity as reported in the READ CAPACITY (10) parameter data (see 5.15.2) and the READ CAPACITY (16) parameter data (see 5.16.2). The device server shall not cause any LBA on a fully provisioned logical unit to become unmapped (i.e., anchored or deallocated).

A fully provisioned logical unit does not support logical block provisioning management (see 4.7.3). A fully provisioned logical unit may support the GET LBA STATUS command (see 5.4).

The device server in a fully provisioned logical unit shall set the LBPME bit to zero in the READ CAPACITY (16) parameter data (see 5.16.2).

### 4.7.3 Logical block provisioning management

#### 4.7.3.1 Logical block provisioning management overview

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

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

The device server in a thin provisioned logical unit may report more LBAs in the READ CAPACITY (10) parameter data (see 5.15.2) and the READ CAPACITY (16) parameter data (see 5.16.2) than the number of mapped LBAs in the logical unit.

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

The device server in a logical unit that supports logical block provisioning management sets the LBPME bit to one in the READ CAPACITY (16) parameter data (see 5.16.2).

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

a) shall support the Logical Block Provisioning VPD page (see 6.6.4);
b) may supply a provisioning group designation descriptor as defined in the Logical Block Provisioning VPD page; and
c) may support logical block provisioning thresholds (see 4.7.3.8.1).

The device server in a logical unit that supports logical block provisioning management shall support at least one of the following unmap mechanisms:

a) the UNMAP command (see 5.28);
b) the UNMAP bit in the WRITE SAME (10) command (see 5.43); or
c) the UNMAP bit in the WRITE SAME (16) command (see 5.44).

A device server that supports the UNMAP command should support the Block Limits VPD page. If a device server supports the UNMAP command and the Block Limits VPD page, then the device server shall:

a) set the MAXIMUM UNMAP LBA COUNT field in the Block Limits VPD page (see 6.6.3) to a value greater than or equal to one; and
b) set the MAXIMUM UNMAP DESCRIPTOR COUNT field in the Block Limits VPD page to a value greater than or equal to one.

The device server in a logical unit that supports logical block provisioning management may support the GET LBA STATUS command (see 5.4).

If the device server supports:

a) the UNMAP bit in the WRITE SAME (10) command or in the WRITE SAME (16) command; and
b) the WRITE SAME (32) command (see 5.45),

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

#### 4.7.3.2 Resource provisioned logical unit

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

Every LBA in a resource provisioned logical unit shall be either mapped or anchored (i.e., not deallocated). A resource provisioned logical unit shall provide LBA mapping resources sufficient to map all logical blocks for the logical unit's capacity as reported in the READ CAPACITY (10) parameter data (see 5.15.2) and the READ CAPACITY (16) parameter data (see 5.16.2). A resource provisioned logical unit may provide resources in excess of this requirement. The device server shall not cause any LBA on a resource provisioned logical unit to become deallocated.

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

a) the LBPME bit to one in the READ CAPACITY (16) parameter data (see 5.16.2);

b) the PROVISIONING TYPE field to 001b (i.e., resource provisioned) in the Logical Block Provisioning VPD page (see 6.6.4); and
c) the ANC_SUP bit to one in the Logical Block Provisioning VPD page.

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

#### 4.7.3.3 Thin provisioned logical unit

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

Every LBA in a thin provisioned logical unit shall be either mapped, anchored, or deallocated. A thin provisioned logical unit is not required to provide LBA mapping resources sufficient to map all logical blocks for the logical unit's capacity as indicated in the READ CAPACITY (10) parameter data (see 5.15.2) and the READ CAPACITY (16) parameter data (see 5.16.2).

If a logical unit does not support anchored LBAs, then every unmapped LBA in the logical unit shall be deallocated. If a device server in a logical unit that does not support anchored LBAs receives a command to anchor an LBA, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

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

a) the LBPME bit to one in the READ CAPACITY (16) parameter data (see 5.16.2); and
b) the PROVISIONING TYPE field to 010b (i.e., thin provisioned) in the Logical Block Provisioning VPD page (see 6.6.4).

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

#### 4.7.3.4 Unmapping LBAs

##### 4.7.3.4.1 Processing unmap requests

Application clients may use an UNMAP command (see 5.28) or a WRITE SAME command (see 5.43, 5.44, and 5.45) to request that LBAs be unmapped. For each LBA that is requested to be unmapped, the device server shall:

a) perform an unmap operation (see 4.7.3.4.2) on the LBA; or
b) make no change to the logical block provisioning state of the LBA.

The application client may determine the provisioning state of LBAs using the GET LBA STATUS command (see 5.4).

##### 4.7.3.4.2 Unmap operations

An unmap operation:

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

The data in all other mapped LBAs on the medium shall be preserved. Performing an unmap operation on an unmapped LBA shall not be considered an error.

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

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

##### 4.7.3.4.3 WRITE SAME command and unmap operations

A WRITE SAME command (see 5.43, 5.44, and 5.45) may be used to request unmap operations. A WRITE SAME command shall not cause an LBA to become unmapped if unmapping that LBA creates a case in which a subsequent read of that unmapped LBA may return user data or protection information that differs from the Data-Out Buffer for that WRITE SAME command. The protection information returned by a read of an

unmapped LBA is set to FFFF_FFFF_FFFF_FFFFh (see 4.7.3.6).

If the device server does not support allowing a WRITE SAME command to request unmap operations, then the device server shall perform the write medium operation based on the rules for caching specified by that command on the specified LBAs, and this shall not be considered an error.

If a device server:

a) receives a WRITE SAME command with the UNMAP bit set to one;
b) supports using the received command to request an unmap operation; and
c) does not perform an unmap operation (see 4.7.3.4.2) on one or more of the specified LBAs,

then:

a) for each specified LBA for which an unmap operation was not performed, the device server shall perform the write medium operation based on the rules for caching specified by that command; and
b) this shall not be considered an error.

The device server shall perform the write medium operation based on the rules for caching specified by a WRITE SAME command, and shall not perform any unmap operations, if the device server sets the LBPRZ bit to one in the READ CAPACITY (16) parameter data (see 5.16.2), and:

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

#### 4.7.3.5 Autonomous LBA transitions

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

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

If the LBPRZ bit in the READ CAPACITY (16) parameter data (see 5.16.2) is set to one, and a mapped LBA references a logical block that contains:

a) user data with all bits set to zero; and
b) protection information, if any, set to FFFF_FFFF_FFFF_FFFFh,

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

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

#### 4.7.3.6 Logical block provisioning management and protection information

If protection information is enabled, then the protection information for LBAs in:

a) the mapped state shall be as described in 4.7.4.3;
b) the deallocated state shall be as described in 4.7.4.4; and
c) the anchored state shall be as described in 4.7.4.5.

#### 4.7.3.7 Resource exhaustion considerations

If:

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

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

If:

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

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

A logical block provisioning threshold may be available to monitor the availability of LBA mapping resources (see 4.7.3.8). A logical block provisioning log parameter that reports available LBA mapping resources may be available in the Logical Block Provisioning log page (see 6.4.4).

The resource exhaustion conditions described in this subclause shall only apply to thin provisioned logical units.

#### 4.7.3.8 Logical block provisioning thresholds

##### 4.7.3.8.1 Logical block provisioning thresholds overview

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

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

a) shall support the Logical Block Provisioning mode page (see 6.5.7); and
b) may support the Logical Block Provisioning log page (see 6.4.4).

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

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

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

where:

| | |
|---|---|
| threshold minimum | is the lowest number of LBAs in the range for this threshold |
| threshold maximum | is the highest number of LBAs in the range for this threshold |
| threshold count | is the center of the threshold range for this threshold (i.e., the threshold count value as specified in the threshold descriptor in the Logical Block Provisioning mode page) |
| threshold set size | is the number of LBAs in each threshold set (i.e., $2^{(\text{threshold exponent})}$ indicated in the Logical Block Provisioning VPD page (see 6.6.4)) |

Table 4 defines the meaning of the combinations of values for the THRESHOLD RESOURCE field, the THRESHOLD TYPE field, and the THRESHOLD ARMING field that are used for logical block provisioning thresholds. See the Logical Block Provisioning mode page for the definition of these fields.

**Table 4 — THRESHOLD RESOURCE, THRESHOLD TYPE, and THRESHOLD ARMING values for logical block provisioning thresholds**

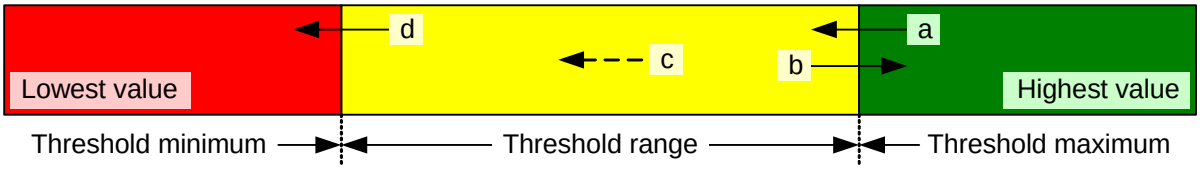
| THRESHOLD RESOURCE value | THRESHOLD TYPE value | THRESHOLD ARMING value | Description |
|---|---|---|---|
| 01h | 000b | 000b | The device server applies the threshold to the availability of LBA mapping resources and performs notifications as the availability of those resources decreases. [a] |
| 02h | 000b | 001b | The device server applies the threshold to the usage of LBA mapping resources and performs notifications as the usage of those resources increases. |
| All other combinations | | | Reserved |

[a] The point when availability of LBA mapping resources reaches zero, corresponds to the persistent lack of LBA mapping resources described in 4.7.3.7.

#### 4.7.3.8.2 Logical block provisioning armed decreasing thresholds

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

If enabled, reporting of armed decreasing threshold events (i.e., the THRESHOLD ARMING field in the threshold descriptor in the Logical Block Provisioning mode page is set to 000b (see 6.5.7)) operates as shown in figure 7.




Notes:

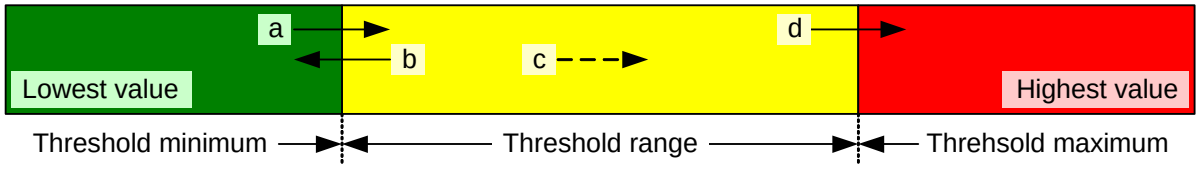
a) if the value to which the threshold is being applied drops below the threshold maximum for the threshold range, then the notification trigger shall be enabled;
b) if the value to which the threshold is being applied increases above the threshold maximum for the threshold range, then the notification trigger shall be disabled;
c) if the notification trigger is enabled, then the device server may disable the notification trigger and perform logical block provisioning threshold notification (see 4.7.3.8.4); and
d) if the notification trigger is enabled and the value to which the threshold is being applied drops below the threshold minimum for the threshold range, then the device server shall disable the notification trigger and perform logical block provisioning threshold notification as defined in 4.7.3.8.4.

**Figure 7 — Armed decreasing threshold operation**

#### 4.7.3.8.3 Logical block provisioning armed increasing thresholds

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

If enabled, reporting of armed increasing threshold events (i.e., the THRESHOLD ARMING field in the threshold descriptor in the Logical Block Provisioning mode page is set to 000b (see 6.5.7)) operates as shown in figure 8.




Notes:

a) if the value to which the threshold is being applied increases above the threshold minimum for the threshold range, then the notification trigger shall be enabled;
b) if the value to which the threshold is being applied decreases below the threshold minimum for the threshold range, then the notification trigger shall be disabled;
c) if the notification trigger is enabled, then the device server may disable the notification trigger and perform logical block provisioning threshold notification (see 4.7.3.8.4); and
d) if the notification trigger is enabled and the value to which the threshold is being applied increases above the threshold maximum for the threshold range, then the device server shall disable the notification trigger and perform logical block provisioning threshold notification as defined in 4.7.3.8.4.

**Figure 8 — Armed increasing threshold operation**

#### 4.7.3.8.4 Logical block provisioning threshold notification

If the LBPERE bit in the Read-Write Error Recovery mode page (see 6.5.8) is set to one, then logical block provisioning threshold notification is enabled and the device server shall perform notification for thresholds with the THRESHOLD TYPE field in the threshold descriptor in the Logical Block Provisioning mode page set to 000b (see 6.5.7) as follows:

a) if the SITUA bit in the Logical Block Provisioning mode page is set to one, then:
   A) if the device server has not established a unit attention condition as a result of this threshold being crossed since the last logical unit reset (see SAM-5), then the device server shall establish a unit attention condition with the additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED for only the initiator port associated with the I_T nexus on which the command was received; or
   B) if the device server has established a unit attention condition as a result of this threshold being crossed since the last logical unit reset, then the device server should establish a unit attention condition with the additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED for only the initiator port associated with the I_T nexus on which the command was received unless establishment of the unit attention condition causes a vendor specific frequency of unit attention conditions for this threshold to be exceeded;

   or

b) if the SITUA bit is set to zero, then:
   A) if the device server has not established a unit attention condition with the initiator port associated with all I_T nexuses as a result of this threshold being crossed since the last logical unit reset (see SAM-5), then the device server shall establish a unit attention condition with the additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED for the initiator port associated with every I_T nexus; or

B) if the device server has established a unit attention condition with the initiator ports associated with all I_T nexuses as a result of this threshold being crossed since the last logical unit reset, then the device server should establish a unit attention condition with the additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED for the initiator port associated with every I_T nexus, unless establishment of the unit attention condition causes a vendor specific frequency of unit attention conditions for this threshold to be exceeded.

If a unit attention condition is established, then, in the returned sense data (see 4.18.1), the VALID bit shall be set to one, and the byte offset in the Logical Block Provisioning mode page of the first byte of the threshold descriptor to which this threshold notification applies shall be returned in the INFORMATION field.

If a unit attention with the additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED is received by the application client, then the application client should reissue the command and take further recovery actions (e.g., administrator notification or other administrator actions). These recovery actions are outside the scope of this standard.

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

The additional sense code THIN PROVISIONING SOFT THRESHOLD REACHED is applicable to both thin provisioned logical units and resource provisioned logical units.

### 4.7.4 LBP (logical block provisioning) state machine

#### 4.7.4.1 LBP state machine overview

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

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

This state machine consists of the following states:

a) LBP1:Mapped state (see 4.7.4.3);
b) LBP2:Deallocated state (see 4.7.4.4); and
c) LBP3:Anchored state (see 4.7.4.5).

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

a) LBP3:Anchored for a resource provisioned logical unit; and
b) LBP2:Deallocated for a thin provisioned logical unit.

If the logical unit does not support anchored LBAs, then the LBP3:Anchored state is not present in the LBP state machine, and the device server shall set the ANC_SUP bit to zero in the Logical Block VPD page (see 6.6.4).

If the logical unit does not support deallocated LBAs, then the LBP2:Deallocated state is not present in the LBP state machine, and the device server shall set the ANC_SUP bit to one.

Figure 9 describes the LBP state machine.

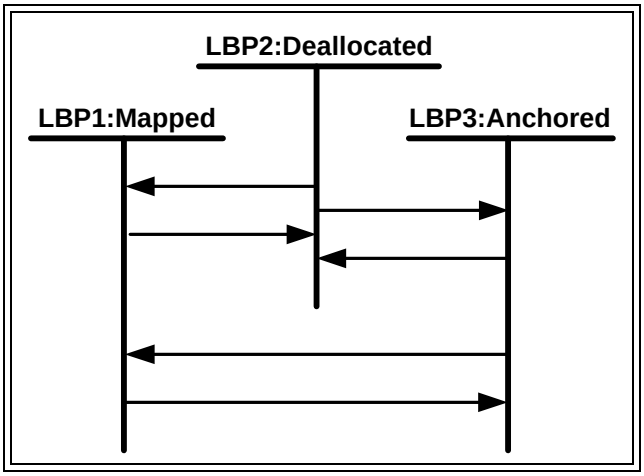



**Figure 9 — LBP state machine**

#### 4.7.4.2 Processing read commands

Table 5 defines the logical block data that a read command shall return for a mapped LBA.

**Table 5 — Logical block data returned by a read command for a mapped LBA**

| Condition | Logical block data returned |
|---|---|
| The LBA became mapped as the result of a format operation or sanitize operation and no write command has specified that LBA since the LBA became mapped. | The logical block data that was written to that LBA by the format operation or the sanitize operation. |
| The LBA became mapped as the result of a write command and no additional write command has specified that LBA since the LBA was mapped. | The logical block data that was written to that LBA by that write command. |
| The LBA became mapped as the result of an autonomous transition, and no write command has specified that LBA since the LBA was mapped. | The logical block data that would be returned if that autonomous transition had not occurred and the LBA had remained unmapped (see table 6). |
| A write command has specified that LBA since that LBA was mapped. | The logical block data that was most recently written to that LBA. |

Table 6 defines the logical block data that a read command shall return for an unmapped LBA.

**Table 6 — Logical block data returned by a read command for an umapped LBA**

| LBPRZ bit [a] | Method used to unmap the LBA | Logical block data returned |
|---|---|---|
| 0 | Write-or-unmap command [b] | The value that would have been written if the write had been chosen. |
| | Not a write-or-unmap command [c] | Logical block data containing:<br>a) user data set to a vendor-specific value that is not obtained from any other LBA; and<br>b) protection information, if any, set to FFFF_FFFF_FFFF_FFFFh. |
| 1 | Any | Logical block data containing:<br>a) user data set to zero; and<br>b) protection information, if any, set to FFFF_FFFF_FFFF_FFFFh. |

[a] The LBPRZ bit is in the READ CAPACITY (16) parameter data (see 5.16.2).

[b] A write-or-unmap command is a command for which the device server is allowed to perform either:
a) a write with specified logical block data; or
b) an unmap operation if the logical block data returned by read commands for unmapped LBAs matches the specified logical block data.

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

[c] These methods include but are not limited to:
a) a FORMAT UNIT command not specifying an initialization pattern;
b) a REASSIGN BLOCKS command;
c) a SANITIZE command specifying a sanitize block erase operation or a sanitize cryptographic erase operation;
d) an UNMAP command; and
e) an autonomous transition.

After a read command returns a value for an LBA, subsequent read commands for that LBA shall return the same value until a subsequent write command or unmap operation for that LBA is completed or terminated (see 4.14) by the device server.

#### 4.7.4.3 LBP1:Mapped state

##### 4.7.4.3.1 LBP1:Mapped state description

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

If this state was entered from the LBP2:Deallocated state (see 4.7.4.4), then the device server shall allocate LBA mapping resources, if any, required to map the LBA. If this state was entered from the LBP3:Anchored state (see 4.7.4.5), then the device server does not allocate LBA mapping resources.

The device server shall process a read command specifying an LBA in this state (i.e., a mapped LBA) as described in 4.7.4.2.

##### 4.7.4.3.2 Transition LBP1:Mapped to LBP2:Deallocated

This transition shall occur after an unmap operation that transitions a mapped LBA to the deallocated state completes without error (see 4.7.3.4).

If the LBPRZ bit in the READ CAPACITY (16) parameter data (see 5.16.2) is set to one, and the mapped LBA references a logical block that contains:

a) user data with all bits set to zero; and
b) protection information, if any, set to FFFF_FFFF_FFFF_FFFFh,

then this transition may occur at any time.

#### 4.7.4.3.3 Transition LBP1:Mapped to LBP3:Anchored

This transition shall occur after an unmap operation that transitions a mapped LBA to the anchored state completes without error (see 4.7.3.4).

This transition shall not cause the resource exhaustion conditions described in 4.7.3.7.

If the LBPRZ bit in the READ CAPACITY (16) parameter data (see 5.16.2) is set to one, and the mapped LBA references a logical block that contains:

a) user data with all bits set to zero; and
b) protection information, if any, set to FFFF_FFFF_FFFF_FFFFh,

then this transition may occur at any time.

### 4.7.4.4 LBP2:Deallocated state

#### 4.7.4.4.1 LBP2:Deallocated state description

Upon entry into this state, the relationship between the LBA and any physical block(s) shall be removed and the device server should deallocate LBA mapping resources that are no longer in use.

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

The device server shall process a read command specifying an LBA in this state (i.e., a deallocated LBA) as described in 4.7.4.2.

#### 4.7.4.4.2 Transition LBP2:Deallocated to LBP1:Mapped

This transition:

a) shall occur after a write command specifying a deallocated LBA is completed by the device server without error; or
b) may occur at any other time.

#### 4.7.4.4.3 Transition LBP2:Deallocated to LBP3:Anchored

This transition:

a) shall occur after an unmap operation that transitions a deallocated LBA to the anchored state is completed by the device server without error (see 4.7.3.4); or
b) may occur at any other time.

### 4.7.4.5 LBP3:Anchored state

#### 4.7.4.5.1 LBP3:Anchored state description

Upon entry into this state:

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

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

If this state was entered from the LBP1:Mapped state, then the device server does not allocate LBA mapping resources, and the device server relies on LBA mapping resources already allocated to the LBA.

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

The device server shall process a read command specifying an LBA in this state (i.e., an anchored LBA) as described in 4.7.4.2.

#### 4.7.4.5.2 Transition LBP3:Anchored to LBP1:Mapped

This transition shall not cause the resource exhaustion conditions described in 4.7.3.7, and:

a) shall occur after a write command specifying an anchored LBA is completed by the device server without error; or
b) may occur at any other time.

#### 4.7.4.5.3 Transition LBP3:Anchored to LBP2:Deallocated

This transition shall occur after an unmap operation that transitions an anchored LBA to the deallocated state completes without error (see 4.7.3.4).

### 4.8 Data de-duplication

Data de-duplication is the ability of a device server to recognize redundant or duplicate data and reduce the number of duplicate or redundant copies of the data while maintaining the application client supplied LBA of the duplicate or redundant copies of the data. De-duplication shall not affect protection information, if any. Logical units that support data de-duplication may report a count of LBA resources that have been made available as a result of being de-duplicated (see 6.4.4.5).

### 4.9 Ready state

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

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

Some direct-access block devices may be switched from being ready to being not ready by using the START STOP UNIT command (see 5.25). An application client may need to issue a START STOP UNIT command with a START bit set to one to make a direct-access block device ready.

### 4.10 Initialization

A direct-access block device may require initialization of its medium prior to processing logical block access commands. This initialization is requested by an application client using a FORMAT UNIT command (see 5.3). Parameters related to the format (e.g., logical block length) may be set with a MODE SELECT command (see SPC-4 and 6.5.2) prior to the format operation. Some direct-access block devices are initialized by means not specified in this standard. The time when the initialization occurs is vendor specific.

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

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

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

A direct-access block device may become format corrupt (e.g., a change in the number of logical blocks results in ranges of logical blocks with different logical block lengths or multiple protection types) after

processing a MODE SELECT command that changes parameters related to the medium format. During this time, the device server may terminate medium access commands with CHECK CONDITION status with the sense key set to NOT READY and the appropriate additional sense code for the condition.

Any time the READ CAPACITY (10) parameter data (see 5.15.2) and the READ CAPACITY (16) parameter data (see 5.16.2) changes (e.g., when a FORMAT UNIT command or a MODE SELECT command completes changing the number of logical blocks, logical block length, protection information, or reference tag ownership values, or when a vendor specific mechanism causes a change), then the device server shall establish a unit attention condition for the SCSI initiator port (see SAM-5) associated with each I_T nexus, except the I_T nexus on which the command causing the change was received with the additional sense code set to CAPACITY DATA HAS CHANGED.

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

## 4.11 Sanitize operations

### 4.11.1 Sanitize operations overview

A sanitize operation causes the device server to:

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

One of the following sanitize operations may be requested:

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

An application client may request that a sanitize operation be processed in the restricted completion mode or the unrestricted completion mode (see 4.11.3) using the AUSE bit in the SANITIZE command CDB (see 5.24.1).

In the unrestricted completion mode, a SANITIZE command with the EXIT FAILURE MODE service action clears a failed sanitize operation.

In the restricted completion mode the only method to clear a failed sanitize operation is another sanitize operation. If a sanitize operation in the restricted completion mode completes with an error, and a subsequent SANITIZE command requests the unrestricted completion mode, then the device server shall terminate that SANITIZE command with CHECK CONDITION status sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB (see 5.24.1).

All sanitize operations shall be performed on:

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

An application client requests that the device server perform a sanitize operation using the SANITIZE command (see 5.24). If the medium is write protected (see 4.12), then the device server shall terminate a SANITIZE command with CHECK CONDITION status with the sense key set to DATA PROTECT.

### 4.11.2 Processing a sanitize operation

Before processing a sanitize operation, the device server shall:

a) terminate all commands in all task sets except INQUIRY commands, REPORT LUNS commands, and REQUEST SENSE commands with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, SANITIZE IN PROGRESS;
b) stop all enabled power condition timers (see SPC-4);
c) stop all timers for enabled background scan operations (see 4.24); and
d) stop all timers or counters enabled for device specific background functions (see SPC-4).

While processing a sanitize operation, the device server shall:

a) terminate all commands, except INQUIRY commands, REPORT LUNS commands, and REQUEST SENSE commands, with CHECK CONDITION status with the sense key set to NOT READY, the additional sense code set to LOGICAL UNIT NOT READY, SANITIZE IN PROGRESS, and the progress indication field in the sense data set to indicate the progress of the sanitize operation;
b) process all INQUIRY commands, REPORT LUNS commands, and REQUEST SENSE commands;
c) provide pollable sense data (see SPC-4) with the sense key set to NOT READY, the additional sense code set to LOGICAL UNIT NOT READY, SANITIZE IN PROGRESS, and the progress indication field set to indicate the progress of the sanitize operation;
d) ignore all SCSI events (see SAM-5), except a power on event;
e) resume processing the sanitize operation after processing a power on condition;
f) process task management functions without affecting the processing of the sanitize operation;
g) not alter mode data, INQUIRY data, or READ CAPACITY (16) parameter data (e.g., the number of logical blocks, logical block length, or protection information settings for the logical unit); and
h) identify inaccessible physical blocks in a manner that prevents future access to these blocks following a successful sanitize operation.

### 4.11.3 Completing a sanitize operation

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

a) the LBAs should be in the initial condition defined in 4.7.3.2 or 4.7.3.3; and
b) read medium operations and write medium operations should complete without error (see 4.7.4.4.1 and 4.7.4.5.1).

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

a) read medium operation completion status depends on the type of sanitize operation that was performed (see 5.24.2.2, 5.24.2.3 and 5.24.2.4); and
b) write medium operations should complete without error.

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

a) terminate the SANITIZE command being processed, if any (e.g., the IMMED bit was set to zero in the CDB, and the failure occurs before status is returned for the command), with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to SANITIZE COMMAND FAILED; and
b) until completion of a successful sanitize operation has occurred, terminate all commands, except SANITIZE commands allowed in the restricted completion mode, INQUIRY commands, REPORT LUNS commands, and REQUEST SENSE commands, with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to SANITIZE COMMAND FAILED.

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

a) terminate the SANITIZE command being processed, if any (e.g., the IMMED bit was set to zero in the CDB, and the failure occurs before status is returned for the command), with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to SANITIZE COMMAND FAILED; and
b) until completion of a successful sanitize operation has occurred, terminate all commands, except SANITIZE commands, INQUIRY commands, REPORT LUNS commands, and REQUEST SENSE commands, with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to SANITIZE COMMAND FAILED.

A sanitize operation that completed with error and was cleared with a SANITIZE command with the service action of EXIT SANITIZE may have not performed a complete sanitize operation.

After the sanitize operation completes the device server shall:

1) initialize all enabled timers and counters; and
2) start all enabled timers and counters for power conditions and background functions.

### 4.12 Write protection

Write protection prevents the alteration of the medium by commands issued to the device server. Write protection is usually controlled by the user of the medium through manual intervention (e.g., a mechanical lock on the SCSI target device) or may result from hardware controls (e.g., tabs on the medium's housing) or software write protection. All sources of write protection are independent. When present, any write protection shall cause otherwise valid commands that request alteration of the medium to be terminated by the device server with CHECK CONDITION status with the sense key set to DATA PROTECT. Only when all write protections are disabled shall the device server process commands that request alteration of the medium, unmap operations (see 4.7.3.4.2), or sanitize operations (see 4.11).

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

Software write protection results when the device server is marked as write protected by the application client using the SWP bit in the Control mode page (see SPC-4). Software write protection is optional. Changing the state of software write protection shall not prevent previously accepted data (e.g., data in cache) from being written to the medium.

The device server reports the status of write protection in the device server and on the medium with the device-specific parameter field in the mode parameter header (see 6.5.1).

### 4.13 Medium defects

#### 4.13.1 Medium defects overview

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

A medium defect causes:

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

where the logical unit's recovery limits are:

a) specified in the Read-Write Error Recovery mode page (see 6.4.8);

b) specified in the Verify Error Recovery mode page (see 6.4.9); or
c) vendor specific if the device server does not implement the Read-Write Error Recovery mode page or the Verify Error Recovery mode page.

Direct-access block devices may allow the application client to examine and modify the additional information by using the READ LONG commands and the WRITE LONG commands (see 5.19, 5.20, 5.41, and 5.42). The application client may use the WRITE LONG commands to alter the additional information to test the defect detection logic of the direct-access block device or to emulate a logical block with an unrecovered read error when generating a mirror copy. This may induce a recovered error or an unrecovered error.

Direct-access block devices may allow the application client to use the features of the WRITE LONG commands (see 5.41 and 5.42) to:

a) disable error correction on specific logical blocks or physical blocks;
b) disable automatic reassignment on specific logical blocks or physical blocks; and
c) mark specific logical blocks or physical blocks as containing pseudo unrecovered errors with correction enabled or pseudo unrecovered errors with correction disabled.

These features provide methods for an application client to prevent recovered errors and unrecovered errors from being reported as informational exception conditions and prevent unnecessary reassign operations.

During a self-test operation (see SPC-4) or a background scan operation (see 4.24.1), the device server shall:

a) ignore pseudo unrecovered errors with correction disabled; and
b) process pseudo unrecovered errors with correction enabled.

The device server maintains the defect lists defined in table 7.

**Table 7 — Defect lists (i.e., PLIST and GLIST)**

| Defect list | Source | Content |
|---|---|---|
| PLIST (i.e., primary defect list) | Manufacturer | Address descriptors (see 6.2) for physical blocks that contain permanent medium defects and never contain logical block data. |
| GLIST (i.e., grown defect list) | FORMAT UNIT commands (see 5.3) | Address descriptors for physical blocks detected by the device server to have medium defects during an optional certification process performed during a format operation. |
| | | Address descriptors for physical blocks specified in the FORMAT UNIT parameter list (see 5.3.2). |
| | REASSIGN BLOCKS commands (see 5.21) | Address descriptors for physical blocks referenced by the LBAs specified in the reassign LBA list (see 5.21.2) |
| | Read medium operations | Address descriptors for physical blocks that have been reassigned as the result of automatic read reassignment. |
| | Write medium operations | Address descriptors for physical blocks that have been reassigned as the result of automatic write reassignment. |

The READ DEFECT DATA commands (see 5.17 and 5.18) allow the application client to request that the device server return the PLIST and/or the GLIST.

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

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

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

The direct-access block device automatically reassigns defects if allowed by the Read-Write Error Recovery mode page (see 6.5.8).

The device server does not perform automatic read reassignment for an LBA referencing a logical block on which an unrecovered error has occurred. If the application client is notified by the device server that an unrecovered error occurred (e.g., as indicated by a read command being terminated with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to UNRECOVERED READ ERROR) and:

a) the application client is able to regenerate the logical block data for the LBA (e.g., in a redundancy group (see 4.19.1), the application client regenerates logical block data from the logical block data on the other logical units in the redundancy group) and the AWRE bit is set to one in the Read-Write Error Recovery mode page, then the application client may send a write command with that regenerated logical block data to trigger automatic write reassignment;
b) the application client is able to regenerate the logical block data for the LBA and the AWRE bit is set to zero in the Read-Write Error Recovery mode page, then the application client may:
   1) send a REASSIGN BLOCKS command to perform a reassign operation on the LBA; and
   2) send a write command with that regenerated logical block data;

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

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

### 4.13.2 Generation of defect lists

This standard defines address descriptor formats for describing defects (see 6.2). Table 6 lists the defects that each address descriptor format is capable of describing.

**Table 8 — Address descriptor formats**

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

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

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

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

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

a) the MADS bit shall be set to one in the first address descriptor;
b) the MADS bit shall be set to zero in the second address descriptor;
c) the CYLINDER NUMBER field in the first address descriptor shall be less than the CYLINDER NUMBER field in the second address descriptor;
d) the HEAD NUMBER field in the first address descriptor shall be equal to the HEAD NUMBER field in the second address descriptor;

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

### 4.14 Write and unmap failures

If one or more write commands are have not completed when a power loss occurs (e.g., resulting in a vendor specific command timeout by the application client) or a medium error or hardware error occurs (e.g., because a removable medium was incorrectly unmounted), then any data in the logical blocks referenced by the LBAs specified by any of those commands is indeterminate. Before sending a read command or verify command specifying any LBAs that were specified by one of the write commands that did not complete, the application client should resend that write command. If an application client sends a read command or verify command specifying any LBAs that were specified by one of the write commands that did not complete before resending that write command, then the device server may return old data, new data, vendor-specific data, or any combination thereof for the logical blocks referenced by the specified LBAs.:

If logical block provisioning (see 4.7) is supported, and one or more unmap commands are in the task set and have not completed when a power loss, medium error, or hardware error occurs, then the logical block provisioning state of the LBAs requested to be unmapped by any of those commands may or may not have changed. Before sending a read command or verify command specifying any LBAs that were specified by one of the unmap commands that did not complete, the application client should resend that unmap command.

### 4.15 Caches

Direct-access block devices may implement caches. A cache is an area of temporary storage in the direct-access block device with a fast access time that is used to enhance performance. Cache exists separate from the medium and is not directly accessible by the application client.

Cache stores logical block data.

Cache may be volatile or non-volatile. Volatile caches do not retain data through power cycles. Non-volatile cache memories retain data through power cycles. There may be a limit on the amount of time a non-volatile cache is able to retain data without power.

The power, if any, that allows a non-volatile cache to remain non-volatile may become low enough to prevent the non-volatile cache from remaining non-volatile for a vendor specific minimum time (e.g., the battery voltage becomes too low too sustain cache contents beyond a vendor specific time). If this occurs and the Extended INQUIRY Data VPD page (see SPC-4) indicates that the device server contains non-volatile cache (i.e., NV_SUP bit set to one), then:

a) if the reporting of informational exceptions control warnings is enabled (i.e., the EWASC bit is set to one in the Information Exceptions Control mode page (see 6.5.6)), then the device server shall report the degraded non-volatile cache as specified in the Information Exceptions Control mode page with an additional sense code set to WARNING - DEGRADED POWER TO NON-VOLATILE CACHE; or
b) if the reporting of informational exceptions control warnings is disabled (i.e., the EWASC bit is set to zero in the Information Exceptions Control mode page), then the device server shall establish a unit attention condition (see SAM-5) for the initiator port associated with every I_T nexus with the additional sense code set to WARNING - DEGRADED POWER TO NON-VOLATILE CACHE.

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

If a non-volatile cache becomes volatile, then the device server shall set the REMAINING NON-VOLATILE TIME field to zero in the Non-volatile Cache log page (see 6.4.5).

If non-volatile cache becomes volatile and the Extended INQUIRY Data VPD page (see SPC-4) indicates that the device server contains non-volatile cache (i.e., the NV_SUP bit is set to one) then:

a) if the reporting of informational exceptions control warnings is enabled (i.e., the EWASC bit is set to one in the Information Exceptions Control mode page (see 6.5.6)), then the device server shall report

the change in the cache as specified in the Information Exceptions Control mode page with the additional sense code set to WARNING - NON-VOLATILE CACHE NOW VOLATILE; or

b) if the reporting of informational exceptions control warnings is disabled (i.e., the EWASC bit is set to zero in the Information Exceptions Control mode page), then the device server shall establish a unit attention condition (see SAM-5) for the initiator port associated with every I_T nexus with the additional sense code set to WARNING - NON-VOLATILE CACHE NOW VOLATILE.

If:

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

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

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

While processing write commands, the device server may use the cache to store data that is to be written to the medium at a later time. This is called write-back caching. A write command may complete prior to logical blocks being written to the medium. As a result of using write-back caching there is a period of time when the data may be lost if power to the SCSI target device is lost and a volatile cache is being used or a hardware failure occurs. There is also the possibility of an error occurring during the subsequent write medium operation. If an error occurred during the write command, the error may be reported as a deferred error on a later command. The application client may request that write-back caching be disabled with the Caching mode page (see 6.5.5) to prevent detected write errors from being reported as deferred errors. Even with write-back caching disabled, undetected write errors may occur. The VERIFY commands and the WRITE AND VERIFY commands may be used to detect those errors.

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

During an unmap operation, the device server changes any data in the cache for the LBA unmapped by the operation so that any data transferred by the device server to the Data-In Buffer while processing a subsequent read command reflects the results of the unmap operation (see 4.7.4.4.1 and 4.7.4.5.1).

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

NOTE 4 - This does not mean that stale data is allowed in the cache when the DPO bit is set to one. If a write medium operation accesses the same LBA as a logical block in the cache, then the logical block in the cache is updated with the new write data. If an unmap operation accesses the same LBA as a logical block in the cache, then the logical block in the cache is updated with the new read data or the logical block is removed from the cache.

Application clients may use the force unit access (FUA) bit in the CDB of read commands or write commands to specify that the device server shall access the medium.

if the DPO bit and the FUA bit are both set to one in read commands or write commands, then processing of those commands has the same effect as bypassing the cache.

Application clients may use the force unit access non-volatile cache (FUA_NV) bit in CDBs of read commands or write commands to specify that the device server shall access either a non-volatile cache or the medium.

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

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

a) the PRE-FETCH commands (see 5.8 and 5.9) cause a set of logical blocks requested by the application client to be read into cache for possible future access. The logical blocks fetched are subject to later replacement;
b) the SYNCHRONIZE CACHE commands (see 5.26 and 5.27) force any write data in cache in the requested set of logical blocks to be written to the medium. These commands may be used to ensure that the data is written and any detected errors reported;
c) the Caching mode page (see 6.5.5) provides control of cache behavior.

### 4.16 Implicit HEAD OF QUEUE command processing

Each of the following commands defined by this standard may be processed by the task manager as if it has a HEAD OF QUEUE task attribute (see SAM-5), even if the command is received with a SIMPLE task attribute, an ORDERED task attribute, or no task attribute:

a) the READ CAPACITY (10) command (see 5.15); and
b) the READ CAPACITY (16) command (see 5.16).

The following command defined by this standard shall be processed by the task manager as if it has a HEAD OF QUEUE task attribute, even if the command is received with a SIMPLE task attribute, an ORDERED task attribute, or no task attribute:

a) the SANITIZE command (see 5.24)

See SPC-4 for additional commands subject to implicit HEAD OF QUEUE command processing. See SAM-5 for additional rules on implicit head of queue processing.

### 4.17 Reservations

Reservation restrictions are placed on commands as a result of access qualifiers associated with the type of reservation. See SPC-4 for a description of reservations. The details of commands that are allowed under

what types of reservations are described in table 9.

Commands from I_T nexuses holding a reservation should complete normally. Table 9 specifies the behavior of commands from registered I_T nexuses when a registrants only or all registrants type persistent reservation is present.

For each command, this standard or SPC-4 defines the conditions that result in the device server completing the command with RESERVATION CONFLICT status.

**Table 9 — SBC-3 commands that are allowed in the presence of various reservations** (part 1 of 2)

| Command | Addressed logical unit has this type of persistent reservation held by another I_T nexus | | | | |
|---|---|---|---|---|---|
| | From any I_T nexus | | From registered I_T nexus (RR all types) | From I_T nexus not registered | |
| | Write Exclusive | Exclusive Access | | Write Exclusive - RR | Exclusive Access - RR |
| COMPARE AND WRITE | Conflict | Conflict | Allowed | Conflict | Conflict |
| FORMAT UNIT | Conflict | Conflict | Allowed | Conflict | Conflict |
| GET LBA STATUS | Allowed | Conflict | Allowed | Allowed | Conflict |
| ORWRITE (16) / (32) | Conflict | Conflict | Allowed | Conflict | Conflict |
| POPULATE TOKEN | Allowed | Conflict | Allowed | Allowed | Conflict |
| PRE-FETCH (10) / (16) | Allowed | Conflict | Allowed | Allowed | Conflict |
| PREVENT ALLOW MEDIUM REMOVAL (Prevent=0) | Allowed | Allowed | Allowed | Allowed | Allowed |
| PREVENT ALLOW MEDIUM REMOVAL (Prevent <> 0) | Conflict | Conflict | Allowed | Conflict | Conflict |
| READ (10) / (12) / (16) / (32) | Allowed | Conflict | Allowed | Allowed | Conflict |
| READ CAPACITY (10) / (16) | Allowed | Allowed | Allowed | Allowed | Allowed |
| READ DEFECT DATA (10) / (12) | Allowed [a] | Conflict | Allowed | Allowed [a] | Conflict |
| READ LONG (10) / (16) | Conflict | Conflict | Allowed | Conflict | Conflict |
| REASSIGN BLOCKS | Conflict | Conflict | Allowed | Conflict | Conflict |
| RECEIVE ROD TOKEN INFORMATION | See SPC-4 | | | | |
| REPORT REFERRALS | Allowed | Allowed | Allowed | Allowed | Allowed |
| SANITIZE | Conflict | Conflict | Allowed | Conflict | Conflict |

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

[a] The device server in logical units claiming compliance with previous versions of this standard (e.g., SBC-2) may complete the command with RESERVATION CONFLICT status. Device servers may report whether certain commands are allowed in the PERSISTENT RESERVE IN command REPORT CAPABILITIES service action parameter data ALLOW COMMANDS field (see SPC-4).

**Table 9 — SBC-3 commands that are allowed in the presence of various reservations** (part 2 of 2)

| Command | Addressed logical unit has this type of persistent reservation held by another I_T nexus | | | | |
|---|---|---|---|---|---|
| | From any I_T nexus | | From registered I_T nexus (RR all types) | From I_T nexus not registered | |
| | Write Exclusive | Exclusive Access | | Write Exclusive - RR | Exclusive Access - RR |
| START STOP UNIT with START bit set to one and POWER CONDITION field set to 0h | Allowed | Allowed | Allowed | Allowed | Allowed |
| START STOP UNIT with START bit set to zero or POWER CONDITION field set to a value other than 0h | Conflict | Conflict | Allowed | Conflict | Conflict |
| SYNCHRONIZE CACHE (10) / (16) | Conflict | Conflict | Allowed | Conflict | Conflict |
| UNMAP | Conflict | Conflict | Allowed | Conflict | Conflict |
| VERIFY (10) / (12) / (16) / (32) | Allowed | Conflict | Allowed | Allowed | Conflict |
| WRITE (10) / (12) / (16) / (32) | Conflict | Conflict | Allowed | Conflict | Conflict |
| WRITE AND VERIFY (10) / (12) / (16) / (32) | Conflict | Conflict | Allowed | Conflict | Conflict |
| WRITE LONG (10) / (16) | Conflict | Conflict | Allowed | Conflict | Conflict |
| WRITE SAME (10) / (16) / (32) | Conflict | Conflict | Allowed | Conflict | Conflict |
| WRITE USING TOKEN | Conflict | Conflict | Allowed | Conflict | Conflict |
| XDWRITEREAD (10) / (32) | Conflict | Conflict | Allowed | Conflict | Conflict |
| XPWRITE (10) / (32) | Conflict | Conflict | Allowed | Conflict | Conflict |

Key: RR = Registrants Only or All Registrants

Allowed: Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present should complete normally.

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

[a] The device server in logical units claiming compliance with previous versions of this standard (e.g., SBC-2) may complete the command with RESERVATION CONFLICT status. Device servers may report whether certain commands are allowed in the PERSISTENT RESERVE IN command REPORT CAPABILITIES service action parameter data ALLOW COMMANDS field (see SPC-4).

## 4.18 Error reporting

### 4.18.1 Error reporting overview

If any of the conditions listed in table 10 occur during the processing of a command, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to the specified value and the additional sense code set to the appropriate value for the condition. Some errors may occur after the completion status has already been reported. For such errors, SPC-4 defines a deferred error reporting mechanism. Table 10 lists some error conditions and the applicable sense keys. The list does not provide a complete list of all conditions that may cause CHECK CONDITION status.

**Table 10 — Example error conditions**

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

Direct-access block devices compliant with this standard shall support both the fixed and descriptor formats of sense data (see SPC-4). If fixed format sense data is used but the values to be placed in the sense data INFORMATION field or COMMAND-SPECIFIC INFORMATION field are too large for the fixed format sense data (e.g., an 8-byte LBA), then the VALID bit shall be set to zero.

Table 11 summarizes use of the sense data fields.

**Table 11 — Sense data field usage for direct-access block devices**

| Field | Usage | Reference |
|---|---|---|
| VALID bit and INFORMATION field | READ LONG commands | 5.19 and 5.20 |
| | REASSIGN BLOCKS command | 5.21 |
| | WRITE LONG commands | 5.41 and 5.42 |
| | Any command that accesses the medium, based on the Read-Write Error Recovery mode page | 6.5.8 |
| | Any command that accesses the medium, based on the Verify Error Recovery mode page | 6.5.9 |
| | Any command that is terminated with a logical block provisioning threshold notification | 4.7.3.8.4 |
| | COMPARE AND WRITE command | 5.2 |
| COMMAND-SPECIFIC INFORMATION field | EXTENDED COPY command | SPC-4 |
| | REASSIGN BLOCKS command | 5.21 |
| ILI bit | READ LONG commands | 5.19 and 5.20 |
| | WRITE LONG commands | 5.41 and 5.42 |

When a command attempts to access or reference an invalid LBA, the device server shall return the first invalid LBA in the INFORMATION field of the sense data (see SPC-4).

When a recovered read error is reported, the INFORMATION field of the sense data shall contain the last LBA on which a recovered read error occurred for the command.

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

### 4.18.2 Processing pseudo unrecovered errors

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

a) perform no error recovery on the affected logical blocks, including any read error recovery enabled by the Read-Write Error Recovery mode page (see 6.5.8) or the Verify Error Recovery mode page (see 6.5.9);
b) perform no automatic read reassignment or automatic write reassignment for the affected logical blocks, regardless of the settings of the AWRE bit and the ARRE bit in the Read-Write Error Recovery mode page;
c) not consider errors on the affected logical blocks to be informational exception conditions as defined in the Information Exceptions Control mode page (see 6.5.6);
d) not log errors on the affected logical blocks in the Error Counter log pages (see SPC-4); and
e) in any information returned for the error (e.g., in sense data or in the Background Scan Results log page (see 6.4.2)), set the sense key to MEDIUM ERROR and either:
    A) should set the additional sense code to READ ERROR – LBA MARKED BAD BY APPLICATION CLIENT; or
    B) may set the additional sense code to UNRECOVERABLE READ ERROR.

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

a) a format operation;
b) a reassign operation;
c) a sanitize overwrite operation;

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

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

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

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

a) a read command.

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

a) if enabled by the Read-Write Error Recovery mode page (see 6.5.8) or the Verify Error Recovery mode page (see 6.5.9), perform error recovery on the affected logical blocks;
b) perform no automatic read reassignment or automatic write reassignment for the affected logical blocks, regardless of the settings of the AWRE bit and the ARRE bit in the Read-Write Error Recovery mode page;
c) consider errors on the affected logical blocks to be informational exception conditions as defined in the Information Exceptions Control mode page (see 6.5.6);
d) log errors on the affected logical blocks in the Error Counter log pages (see SPC-4); and
e) in any information returned for the error (e.g., in sense data or in the Background Scan Results log page (see 6.4.2)), set the sense key to MEDIUM ERROR and either:
   A) should set the additional sense code to READ ERROR – LBA MARKED BAD BY APPLICATION CLIENT; or
   B) may set the additional sense code to UNRECOVERABLE READ ERROR.

### 4.18.3 Block commands sense data descriptor

Table 12 defines the block commands sense data descriptor used in descriptor format sense data for direct-access block devices.

**Table 12 — Block commands sense data descriptor format**

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

The DESCRIPTOR TYPE field and the ADDITIONAL LENGTH field are defined in SPC-4 and shall be set to the values shown in table 12.

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

### 4.18.4 User data segment referral sense data descriptor

Table 13 defines the user data segment referral sense data descriptor used in descriptor format sense data for direct-access block devices. The user data segment referral sense data descriptor contains descriptors indicating the user data segment(s) on the logical unit and the SCSI target port groups through which those user data segments may be accessed (see 4.28).

**Table 13 — User data segment referral sense data descriptor format**

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

The DESCRIPTOR TYPE field is defined in SPC-4 and shall be set to the value shown in table 13.

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

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

Each user data segment referral descriptor (see table 14) indicates information identifying:

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

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

Table 14 defines the user data segment referral descriptor.

**Table 14 — User data segment referral descriptor format**

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

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

The FIRST USER DATA SEGMENT LBA field indicates the first LBA of the first user data segment (see 4.28) indicated by this descriptor.

The LAST USER DATA SEGMENT LBA field indicates the last LBA of the last user data segment (see 4.28) indicated by this descriptor.

The target port group descriptor (see table 15) specifies the target port group and the asymmetric access state of the target port group (see SPC-4). The device server shall return one target port group descriptor for each target port group in a target port asymmetric access state of active/optimized, active/non-optimized, or transitioning. The device server may return one target port group descriptor for each target port group in a target port asymmetric access state of unavailable.

**Table 15 — Target port group descriptor**

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

The ASYMMETRIC ACCESS STATE field (see SPC-4) contains the asymmetric access state of the user data segment(s) specified by this descriptor that may be accessed through this target port group.

The TARGET PORT GROUP field specifies a target port group (see SPC-4) that the application client uses when issuing commands associated with the user data segments specified by this descriptor.

## 4.19 Model for XOR commands

### 4.19.1 Model for XOR commands overview

In storage arrays, a storage array controller organizes a group of direct-access block devices into objects. The type of object supported by this model is the redundancy group (see 3.2.73), where some of the logical blocks on the direct-access block devices are used for XOR-protected data (see 3.2.100) and some of the logical blocks are used for check data (see 3.2.13). The check data is generated by performing a cumulative XOR (see 3.2.28) operation of the XOR-protected data. The XOR operation may be performed by the storage array controller or by the direct-access block devices.

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

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

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

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

In contrast, during storage array controller supervised XOR operations (see 4.19.2) only three commands are needed:

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

### 4.19.2 Storage array controller supervised XOR operations

#### 4.19.2.1 Storage array controller supervised XOR operations overview

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

a) update write operation (see 4.19.2.2);
b) regenerate operation (see 4.19.2.3); and
c) rebuild operation (see 4.19.2.4).

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

XDWRITEREAD commands (see 5.47 and 5.48) and XPWRITE commands (see 5.49 and 5.50) are required to implement storage array controller supervised XOR operations. The storage array controller also uses READ commands and WRITE commands for certain operations.

#### 4.19.2.2 Update write operation

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

1) An XDWRITEREAD command is sent to the data disk. This command requests that:
   1) new XOR-protected data be transferred to the data disk. The device server reads the old XOR-protected data, performs an XOR operation using the old XOR-protected data and

the received XOR-protected data, retains the intermediate XOR result, and writes the received XOR-protected data to the medium; and
2) the intermediate XOR data be transferred to the storage array controller;
and
2) An XPWRITE command is sent to the parity disk. This command requests that the device server transfer the intermediate XOR data (i.e., XOR data received as the result of a previous XDWRITEREAD command) to the parity disk. The device server reads the old check data, performs an XOR operation using the old check data and the intermediate XOR data, and writes the result (i.e., the new check data) to the medium.

#### 4.19.2.3 Regenerate operation

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

1) A READ command is sent to the first direct-access block device. This causes the data to be transferred from the direct-access block device to the storage array controller;
2) An XDWRITEREAD command with the DISABLE WRITE bit set to one is sent to the next direct-access block device. This command requests that:
   1) the data from the previous read medium operation be transferred to the next direct-access block device. The direct-access block device reads its data, performs an XOR operation on the received data and its data, and retains the intermediate XOR result; and
   2) the intermediate XOR data be transferred from the device to the storage array controller;
   and
3) Step 2) is repeated until all direct-access block devices in the redundancy group except the failed device have been accessed.

#### 4.19.2.4 Rebuild operation

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

1) A READ command is sent to the first direct-access block device. This transfers the data from the direct-access block device to the storage array controller;
2) An XDWRITEREAD command with the DISABLE WRITE bit set to one is sent to the next direct-access block device. This command requests that:
   1) the data from the previous read medium operation be transferred to the next direct-access block device. The device server reads its data, performs an XOR operation using the received data and its data, and retains the intermediate XOR result; and
   2) the intermediate XOR data be transferred from the device to the storage array controller;
3) Step 2) is repeated until all direct-access block devices in the redundancy group, except the replacement device, have been accessed; and
4) A WRITE command is sent to the replacement device. This transfers the regenerated data from step 2) to the replacement device. The replacement device writes the regenerated data to the medium.

### 4.19.3 Array subsystem considerations

#### 4.19.3.1 Array subsystem considerations overview

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

#### 4.19.3.2 Access to an inconsistent stripe

A stripe is a set of corresponding extents (see 3.2.29) from two or more direct-access block devices.

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

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

If a device server terminates any of the XOR commands 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, the identify the scope of the failure, then limit access to the inconsistent stripe. The recovery procedures that the storage array controller implements are outside the scope of this standard.

## 4.20 Rebuild assist mode

### 4.20.1 Rebuild assist mode overview

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

Enabling the rebuild assist mode:

a) may cause the device server to initiate a self test to identify the scope of failures, if any;
b) modifies READ command recovery behavior by the device server based on the setting of the RARC bit (see 4.20.3 and 5.11); and
c) defines sense data returned by the device server to indicate the location of multiple failing logical blocks on read commands and write commands.

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

### 4.20.2 Enabling rebuild assist mode

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

If the ENABLED bit is set to one in the Rebuild Assist Output diagnostic page and the device server validates the page, then the device server:

a) shall enable the rebuild assist mode;

b) may perform a diagnostic test of the physical elements contained within the logical unit;
c) if a diagnostic test of the physical elements is performed, then should disable any physical elements that are not functioning; and
d) should complete the command without error.

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

### 4.20.3 Using the rebuild assist mode

#### 4.20.3.1 Using rebuild assist mode overview

After rebuild assist mode is enabled, the application client should issue read commands to read the available logical block data from the failed logical unit. If the device server does not detect an unrecovered error while processing a read command, then the device server should complete the read command without error.

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

#### 4.20.3.2 Unpredicted unrecovered read error

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

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

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

If the application client receives sense data with the sense key set to MEDIUM ERROR and the additional sense code set to UNRECOVERED READ ERROR, then the application client should issue the next read command with the starting LBA set to the contents of the INFORMATION field plus one.

#### 4.20.3.3 Predicted unrecovered read error

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

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

1) shall perform limited read recovery that is vendor specific;
2) shall transfer the data for all recovered logical blocks, if any, from the logical block referenced by the starting LBA of the failed read command up to the first predicted unrecovered LBA to the Data-in Buffer; and
3) shall terminate the read command with CHECK CONDITION status with the sense key set to ABORTED COMMAND, the additional sense code set to MULTIPLE READ ERRORS, the INFORMATION field set to the LBA referencing the first unrecovered logical block, and the COMMAND-SPECIFIC INFORMATION field set to the LBA referencing the last unrecovered logical block in a sequence of contiguous unrecovered logical blocks that started with the LBA specified in the INFORMATION field.

If the application client receives sense data with the sense key set to ABORTED COMMAND and the additional sense code set to MULTIPLE READ ERRORS, then the application client should issue the next

read command with the starting LBA set to the contents of the COMMAND-SPECIFIC INFORMATION field plus one to continue recovering data from the logical unit. This process should be repeated until all of the LBAs have been scanned.

#### 4.20.3.4 Unpredicted unrecovered write error

If the device server detects an unpredicted unrecovered error while processing a write command, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to MEDIUM ERROR, the additional sense code set to WRITE ERROR, and the INFORMATION field set to the LBA referencing the first logical block in error.

#### 4.20.3.5 Predicted unrecovered write error

If the device server detects a predicted unrecovered error while processing a write command, then the device server:

1) transfers the write data from the Data-out Buffer;
2) writes the transferred data up to the logical block referenced by the failing LBA; and
3) shall terminate the write command with CHECK CONDITION status with the sense key set to ABORTED COMMAND, the additional sense code set to MULTIPLE WRITE ERRORS, the INFORMATION field set to the LBA referencing the first logical block in error, and the COMMAND-SPECIFIC INFORMATION field set to the LBA referencing the last logical block in error in a sequence of contiguous logical blocks that started with the LBA specified in the INFORMATION field.

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

### 4.20.4 Disabling the rebuild assist mode

Rebuild assist mode shall be disabled after a power on.

Rebuild assist mode shall not be affected by a hard reset, an I_T nexus loss, or any task management functions (see SAM-5).

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

### 4.20.5 Testing rebuild assist mode

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

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

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

An application client ends a test by disabling the rebuild assist mode (see 4.20.4).

## 4.21 START STOP UNIT and power conditions

### 4.21.1 START STOP UNIT and power conditions overview

The START STOP UNIT command (see 5.25) allows an application client to control the power condition of a logical unit. This method includes specifying that the logical unit transition to a specific power condition.

In addition to the START STOP UNIT command, the power condition of a logical unit may be controlled by the Power Condition mode page (see SPC-4). If both the START STOP UNIT command and the Power Condition mode page methods are being used to control the power condition of the same logical unit, then the power condition specified by any START STOP UNIT command shall override the Power Condition mode page's power control.

### 4.21.2 Processing of concurrent START STOP UNIT commands

While a START STOP UNIT command is being processed, receipt of a subsequent START STOP UNIT command that requests a different power condition than the power condition requested by the START STOP UNIT command being processed exceeds the capabilities of the SSU_PC state machine (e.g., the state machine is not able to represent changing to the standby_y power condition and the idle_b power condition at the same time).

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

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

### 4.21.3 Managing medium access commands during a change to the active power condition

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

a) polling the power condition using the REQUEST SENSE command (see SPC-4); or
b) sending a START STOP UNIT command with the IMMED bit set to zero and the START bit set to one and waiting for GOOD status to be returned.

### 4.21.4 Stopped Power Condition

In addition to the active power condition, idle power conditions, and standby power conditions described in SPC-4, this standard describes the stopped power condition.

While in the stopped power condition:

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

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

### 4.21.5 START STOP UNIT and power condition state machine

#### 4.21.5.1 START STOP UNIT and power condition state machine overview

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

a) logical unit power states and transitions resulting from specifications in the START STOP UNIT command;
b) settings in the Power Condition mode page (see SPC-4); and
c) the processing of commands.

NOTE 5 - The SSU_PC state machine is an enhanced version of the PC (power condition) state machine described in SPC-4.

The SSU_PC state machine consists of the states shown in table 16.

**Table 16 — Summary of states in the SSU_PC state machine**

| State | Reference | PC state machine state with additional definition (see SPC-4) |
|---|---|---|
| SSU_PC0:Powered_On [a] | 4.21.5.2 | PC0:Powered_On |
| SSU_PC1:Active | 4.21.5.3 | PC1:Active |
| SSU_PC2:Idle | 4.21.5.4 | PC2:Idle |
| SSU_PC3:Standby | 4.21.5.5 | PC3:Standby |
| SSU_PC4:Active_Wait | 4.21.5.6 | PC4:Active_Wait |
| SSU_PC5:Wait_Idle | 4.21.5.7 | PC5:Wait_Idle |
| SSU_PC6:Wait_Standby | 4.21.5.8 | PC6:Wait_Standby |
| SSU_PC7:Idle_Wait | 4.21.5.9 | n/a |
| SSU_PC8:Stopped | 4.21.5.10 | n/a |
| SSU_PC9:Standby_Wait | 4.21.5.11 | n/a |
| SSU_PC10:Wait_Stopped | 4.21.5.12 | n/a |

[a] SSU_PC0:Powered_On is the initial state.

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

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

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

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

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

The SSU_PC state machine shall start in the SSU_PC0:Powered_On state after power on. The SSU_PC state machine shall be configured to transition to the SSU_PC4:Active_Wait state or the SSU_PC8:Stopped state after power on by a mechanism outside the scope of this standard.

This state machine references timers controlled by the Power Condition mode page (see SPC-4) and refers to the START STOP UNIT command (see 5.25).

Figure 10 describes the SSU_PC state machine.

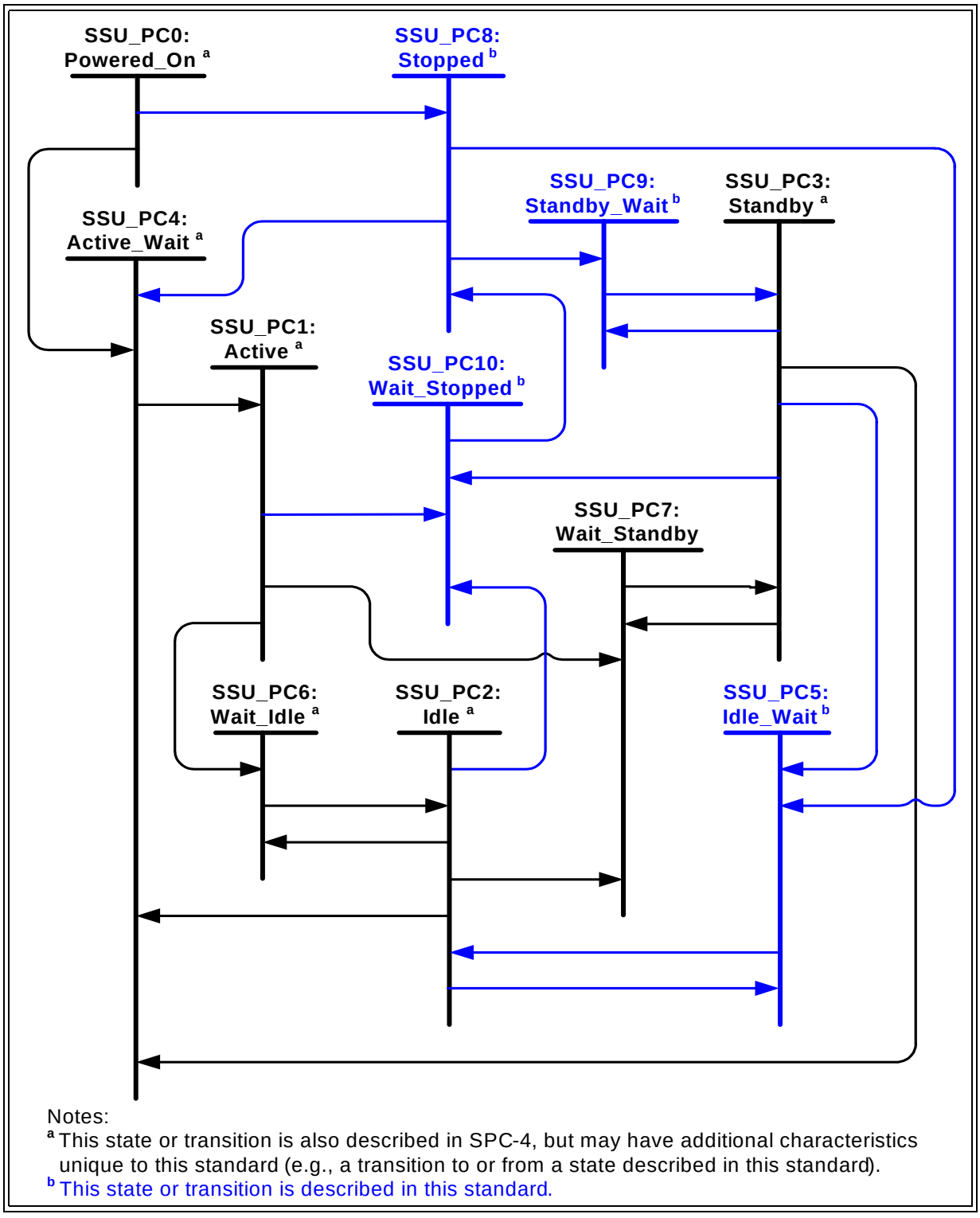



**Figure 10 — SSU_PC state machine**

#### 4.21.5.2 SSU_PC0:Powered_On state

##### 4.21.5.2.1 SSU_PC0:Powered_On state description

See the PC0:Powered_On state in SPC-4 for details about this state.

##### 4.21.5.2.2 Transition SSU_PC0:Powered_On to SSU_PC4:Active_Wait

This transition shall occur if:

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

The transition shall include a Transitioning From Powered On argument.

##### 4.21.5.2.3 Transition SSU_PC0:Powered_On to SSU_PC8:Stopped

This transition shall occur if:

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

The transition shall include a Transitioning From Powered On argument.

#### 4.21.5.3 SSU_PC1:Active state

##### 4.21.5.3.1 SSU_PC1:Active state description

See the PC1:Active state in SPC-4 for details about this state.

##### 4.21.5.3.2 Transition SSU_PC1:Active to SSU_PC5:Wait_Idle

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.25) with the POWER CONDITION field set to 2h (i.e., IDLE); or
b) an idle condition timer (see SPC-4) is enabled, and that timer has expired.

The transition shall include a:

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

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

##### 4.21.5.3.3 Transition SSU_PC1:Active to SSU_PC6:Wait_Standby

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.25) with the POWER CONDITION field set to 3h (i.e., STANDBY); or
b) a standby condition timer (see SPC-4) is enabled and that timer has expired.

The transition shall include a:

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

B) the START STOP UNIT command being processed has the POWER CONDITION MODIFIER field set to 0h (i.e., standby_z power condition);

or

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

#### 4.21.5.3.4 Transition SSU_PC1:Active to SSU_PC10:Wait_Stopped

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.25) with the START bit set to zero and the POWER CONDITION field set to 0h (i.e., START_VALID).

### 4.21.5.4 SSU_PC2:Idle state

#### 4.21.5.4.1 SSU_PC2:Idle state description

See the PC2:Idle state in SPC-4 for details about this state.

#### 4.21.5.4.2 Transition SSU_PC2:Idle to SSU_PC4:Active_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.25) with the START bit set to one and the POWER CONDITION field set to 0h (i.e., START_VALID);
b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to 1h (i.e., ACTIVE); or
c) the device server processes a command that requires the logical unit to be in the SSU_PC1:Active state to continue processing that command.

The transition shall include a:

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

#### 4.21.5.4.3 Transition SSU_PC2:Idle to SSU_PC5:Wait_Idle

This transition shall occur if:

a) the following occur:
   A) an idle condition timer is enabled and that idle condition timer has expired; and
   B) the priority of that idle condition timer is greater than the priority of the idle condition timer associated with the current idle power condition (see SPC-4);

   or
b) the device server processes a START STOP UNIT command (see 5.25) with the POWER CONDITION field set to 2h (i.e., IDLE) and the POWER CONDITION MODIFIER field set to a value that specifies that the logical unit transition to a lower idle power condition.

The transition shall include a:

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

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

#### 4.21.5.4.4 Transition SSU_PC2:Idle to SSU_PC6:Wait_Standby

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.25) with the POWER CONDITION field set to 3h (i.e., STANDBY); or
b) a standby condition timer is enabled and that timer has expired.

The transition shall include a:

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

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

#### 4.21.5.4.5 Transition SSU_PC2:Idle to SSU_PC7:Idle_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.25) with the POWER CONDITION field set to 2h (i.e., IDLE) and the POWER CONDITION MODIFIER field set to a value that specifies that the logical unit transition to a higher idle power condition.

The transition shall include Transitioning From Idle argument and a:

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

#### 4.21.5.4.6 Transition SSU_PC2:Idle to SSU_PC10:Wait_Stopped

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.25) with the START bit set to zero and the POWER CONDITION field set to 0h (i.e., START_VALID).

### 4.21.5.5 SSU_PC3:Standby state

#### 4.21.5.5.1 SSU_PC3:Standby state description

See the PC3:Standby state in SPC-4 for details about this state.

#### 4.21.5.5.2 Transition SSU_PC3:Standby to SSU_PC4:Active_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.25) with the START bit set to one and the POWER CONDITION field set to 0h (i.e., START_VALID);
b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to 1h (i.e., ACTIVE); or
c) the device server processes a command that requires the logical unit to be in the SSU_PC1:Active state to continue processing that command.

The transition shall include a Transitioning From Standby argument.

#### 4.21.5.5.3 Transition SSU_PC3:Standby to SSU_PC6:Wait_Standby

This transition shall occur if:

a) the following occur:
   A) the standby_z condition timer is enabled and that timer expires; and
   B) the priority of that standby condition timer is greater than the priority of the standby condition timer associated with the current standby power condition (see SPC-4);
   or
b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to 3h (i.e., STANDBY) and the POWER CONDITION MODIFIER field set to a value that specifies that the logical unit transition to a lower standby power condition.

The transition shall include Transitioning To Standby_z argument.

#### 4.21.5.5.4 Transition SSU_PC3:Standby to SSU_PC7:Idle_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.25) with the POWER CONDITION field set to 2h (i.e., IDLE); or
b) the device server processes a command and determines that the device server is capable of continuing the processing of that command, when the logical unit is in the SSU_PC2:Idle state.

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

a) Transitioning To Idle_a argument, if:
   A) the device server processes a command and determines that the device server is capable of continuing the processing of that command, when the logical unit is in the idle_a power condition; or
   B) the device server processes a START STOP UNIT command with the POWER CONDITION MODIFIER field set to 0h (i.e., idle_a power condition);
b) Transitioning To Idle_b argument, if:
   A) the device server processes a command and determines that the device server is capable of continuing the processing of that command, when the logical unit is in the idle_b power condition; or
   B) the device server processes a START STOP UNIT command with the POWER CONDITION MODIFIER field set to 1h (i.e., idle_b power condition);
   or
c) Transitioning To_Idle_c argument, if:
   A) the device server processes a command and determines that the device server is capable of continuing the processing of that command, when the logical unit is in the idle_c power condition; or
   B) the device server processes a START STOP UNIT command with the POWER CONDITION MODIFIER field set to 2h (i.e., idle_c power condition).

#### 4.21.5.5.5 Transition SSU_PC3:Standby to SSU_PC9:Standby_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.25) with the POWER CONDITION field set to 3h (i.e., STANDBY) and the POWER CONDITION MODIFIER field set to a value that specifies that the logical unit transition to a higher standby power condition.

The transition shall include a Transitioning To Standby_y argument.

#### 4.21.5.5.6 Transition SSU_PC3:Standby to SSU_PC10:Wait_Stopped

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.25) with the START bit set to zero and the POWER CONDITION field set to 0h (i.e., START_VALID).

#### 4.21.5.6 SSU_PC4:Active_Wait state

##### 4.21.5.6.1 SSU_PC4:Active_Wait state description

While in this state:

a) each idle condition timer that is enabled and not expired is running;
b) each standby condition timer that is enabled and not expired is running;
c) the device server shall provide power management pollable sense data (see SPC-4) with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION; and
d) the logical unit is performing the operations required for it to be in the SSU_PC1:Active state (e.g., a disk drive spins up its medium).

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

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero (see 5.25), that the device server is able to process and complete while in the SSU_PC2:Idle state;
b) the peak power consumed in this state shall be no more than the typical peak power consumed in the SSU_PC1: Active state; and
c) if:
   A) this state was entered with a Transitioning From Idle_c argument; and
   B) the CCF IDLE field in the Power Condition mode page (see SPC-4) is set to 10b (i.e., enabled),

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

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

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero, that the device server is able to process and complete while in the SSU_PC3:Standby state;
b) the peak power consumption in this state is not limited by this standard; and
c) if the CCF STANDBY field in the Power Condition mode page (see SPC-4) is set to 10b (i.e., enabled), then the device server shall terminate any command, except a START STOP UNIT command, that requires the logical unit be in the SSU_PC1:Active state or SSU_PC2:Idle state to continue processing, with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY.

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

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero, that the device server is able to process and complete while in the SSU_PC8:Stopped state;
b) the peak power consumption in this state is not limited by this standard; and
c) if the CCF STOPPED field in the Power Condition mode page (see SPC-4) is set to 10b (i.e., enabled), then the device server shall terminate any TEST UNIT READY command or medium access command, with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY.

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

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

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

#### 4.21.5.6.2 Transition SSU_PC4:Active_Wait to SSU_PC1:Active

See the PC4:Active_Wait to PC1:Active transition in SPC-4 for details about this transition.

### 4.21.5.7 SSU_PC5:Wait_Idle state

#### 4.21.5.7.1 SSU_PC5:Wait_Idle state description

See the PC5:Wait_Idle state in SPC-4 for details about this state.

#### 4.21.5.7.2 Transition SSU_PC5:Wait_Idle to SSU_PC2:Idle

See the PC5:Wait_Idle to PC2:Idle transition in SPC-4 for details about this transition.

### 4.21.5.8 SSU_PC6:Wait_Standby state

#### 4.21.5.8.1 SSU_PC6:Wait_Standby state description

See the PC6:Wait_Standby state in SPC-4 for details about this state.

#### 4.21.5.8.2 Transition SSU_PC6:Wait_Standby to SSU_PC3:Standby

See the PC6:Wait_Standby to PC3:Standby transition in SPC-4 for details about this transition.

### 4.21.5.9 SSU_PC7:Idle_Wait state

#### 4.21.5.9.1 SSU_PC7:Idle_Wait state description

While in this state:

a) each idle condition timer that is enabled and not expired is running;
b) each standby condition timer that is enabled and not expired is running;
c) the device server shall provide power management pollable sense data (see SPC-4) with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION; and
d) the logical unit is performing the operations required for it to be in the SSU_PC2:Idle state ((e.g., a disk drive spins up its medium).

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

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero (see 5.25), that the device server is able to process and complete while in the SSU_PC2:Idle state; and
b) the peak power consumed in this state shall be no more than the typical peak power consumed in the SSU_PC1: Active state.

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

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero, that the device server is able to process and complete while in the SSU_PC3:Standby state;
b) the peak power consumption in this state is not limited by this standard; and
c) the CCF STANDBY field in the Power Condition mode page (see SPC-4) is set to 10b (i.e., enabled), then the device server shall terminate any command, except a START STOP UNIT command, that requires the logical unit be in the SSU_PC1:Active state or SSU_PC2:Idle state to continue processing, with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY.

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

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero, that the device server is able to process and complete while in the SSU_PC8:Stopped state;
b) the peak power consumption in this state is not limited by this standard; and
c) if the CCF STOPPED field in the Power Condition mode page (see SPC-4) is set to 10b (i.e., enabled), then the device server shall terminate any TEST UNIT READY command or medium access command, with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY.

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

#### 4.21.5.9.2 Transition SSU_PC7:Idle_Wait to SSU_PC2:Idle

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

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

### 4.21.5.10 SSU_PC8:Stopped state

#### 4.21.5.10.1 SSU_PC8:Stopped state description

While in this state:

a) the logical unit is in the stopped power condition (see 4.21.1);
b) the idle condition timers and the standby condition timers are disabled;
c) the device server shall provide power management pollable sense data (see SPC-4); and
d) the device server shall terminate each medium access command or TEST UNIT READY command (see SPC-4) as described in 4.21.1.

#### 4.21.5.10.2 Transition SSU_PC8:Stopped to SSU_PC4:Active_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.25) with the START bit set to one and the POWER CONDITION field set to 0h (i.e., START_VALID); or
b) the device server processes a START STOP UNIT command with the POWER CONDITION field set to 1h (i.e., ACTIVE).

The transition shall include a Transitioning From Stopped argument.

#### 4.21.5.10.3 Transition SSU_PC8:Stopped to SSU_PC7:Idle_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.25) with the POWER CONDITION field set to 2h (i.e., IDLE).

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

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

#### 4.21.5.10.4 Transition SSU_PC8:Stopped to SSU_PC9:Standby_Wait

This transition shall occur if:

a) the device server processes a START STOP UNIT command (see 5.25) with the POWER CONDITION field set to STANDBY.

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

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

### 4.21.5.11 SSU_PC9:Standby_Wait state

#### 4.21.5.11.1 SSU_PC9:Standby_Wait state description

While in this state:

a) the device server shall provide power management pollable sense data (see SPC-4) with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION;
b) the peak power consumed in this state shall be no more than the typical peak power consumed in the SSU_PC1:Active state; and
c) the logical unit is performing the operations required for it to be in the SSU_PC3:Standby state ((e.g., a block device is activating circuitry).

If this state was entered with a Transitioning From Standby argument, then the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero (see 5.25), that the device server is able to process and complete in the SSU_PC3:Standby state.

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

a) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero, that the device server is able to process and complete while in the SSU_PC8:Stopped state; and
b) if the CCF STOPPED field in the Power Condition mode page (see SPC-4) is set to 10b (i.e., enabled), then the device server shall terminate any TEST UNIT READY command or medium access command, with CHECK CONDITION status, with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT IS IN PROCESS OF BECOMING READY.

#### 4.21.5.11.2 Transition SSU_PC9:Standby_Wait to SSU_PC3:Standby

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

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

### 4.21.5.12 SSU_PC10: Wait_Stopped state

#### 4.21.5.12.1 SSU_PC10:Wait_Stopped state description

While in this state:

a) the device server shall provide power management pollable sense data (see SPC-4) with the sense key set to NO SENSE and the additional sense code set to LOGICAL UNIT TRANSITIONING TO ANOTHER POWER CONDITION;
b) the device server is capable of processing and completing the same commands, except START STOP UNIT commands with the IMMED bit set to zero (see 5.25), that the device server is able to process and complete in the SSU_PC8:Stopped state;
c) the logical unit is performing the operations required for it to be in the SSU_PC8:Stopped state (e.g., a disk drive spins down its medium); and

d) the device server shall terminate any TEST UNIT READY command or medium access command with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, INITIALIZING COMMAND REQUIRED.

#### 4.21.5.12.2 Transition SSU_PC10:Wait_Stopped to SSU_PC8:Stopped

This transition shall occur when:

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

## 4.22 Protection information model

### 4.22.1 Protection information overview

The protection information model provides for protection of user data while it is being transferred between a sender and a receiver. Protection information is generated at the application layer and may be checked by any object associated with the I_T_L nexus (see SAM-5). Once received, protection information is retained (e.g., written to medium, stored in non-volatile memory, or recalculated on read back) by the device server until overwritten. Power loss, hard reset, logical unit reset, and I_T nexus loss shall have no effect on the retention of protection information.

Support for protection information shall be indicated in the PROTECT bit in the standard INQUIRY data (see SPC-4).

If the logical unit is formatted with protection information, and the EMDP bit is set to one in the Disconnect-Reconnect mode page (see SPC-4), then checking of the logical block reference tag within a service delivery subsystem without accounting for modified data pointers and data alignments may cause false errors when logical blocks are transmitted out of order.

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

### 4.22.2 Protection types

#### 4.22.2.1 Protection types overview

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

The type of protection supported by the logical unit shall be indicated in the spt field in the Extended INQUIRY Data VPD page (see SPC-4). The current protection type shall be indicated in the p_type field in the READ CAPACITY (16) parameter data (see 5.16.2).

An application client may format the logical unit to a specific type of protection using the fmtpinfo field and the protection field usage field in the FORMAT UNIT command (see 5.3).

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

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

a) COMPARE AND WRITE;
b) ORWRITE (16);
c) ORWRITE (32);
d) READ (10);
e) READ (12);
f) READ (16);
g) READ (32);
h) VERIFY (10);
i) VERIFY (12);
j) VERIFY (16);
k) VERIFY (32);

l) WRITE (10);
m) WRITE (12);
n) WRITE (16);
o) WRITE (32);
p) WRITE AND VERIFY (10);
q) WRITE AND VERIFY (12);
r) WRITE AND VERIFY (16);
s) WRITE AND VERIFY (32);
t) WRITE SAME (10);
u) WRITE SAME (16);
v) WRITE SAME (32);
w) XPWRITE (10);
x) XPWRITE (32);
y) XDWRITEREAD (10); and
z) XDWRITEREAD (32).

#### 4.22.2.2 Type 0 protection

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

A logical unit that has been formatted with protection information disabled (see 5.2) or a logical unit that does not support protection information (i.e., the PROTECT bit set to zero in the standard INQUIRY data (see SPC-4)) has type 0 protection.

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

If type 0 protection is enabled and the RDPROTECT field, the WRPROTECT field, the VRPROTECT field, or the ORPROTECT field is set to a zero value, then the following medium commands are invalid and shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

a) READ (32);
b) VERIFY (32);
c) WRITE (32);
d) WRITE AND VERIFY (32); and
e) WRITE SAME (32).

#### 4.22.2.3 Type 1 protection

Type 1 protection:

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

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

a) READ (32);
b) VERIFY (32);
c) WRITE (32);
d) WRITE AND VERIFY (32); and
e) WRITE SAME (32).

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

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

#### 4.22.2.4 Type 2 protection

Type 2 protection:

a) defines the content of each LOGICAL BLOCK GUARD field;
b) does not define the content of any LOGICAL BLOCK APPLICATION TAG field; and
c) defines, except for the first logical block addressed by the command, the content of each lOGICAL BLOCK REFERENCE TAG field.

If type 2 protection is enabled and the RDPROTECT field, the WRPROTECT field, the VRPROTECT field, or the ORPROTECT field is set to a non-zero value, then the following medium commands are invalid and shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

a) COMPARE AND WRITE;
b) ORWRITE (16);
c) ORWRITE (32);
d) READ (10);
e) READ (12);
f) READ (16);
g) VERIFY (10);
h) VERIFY (12);
i) VERIFY (16);
j) WRITE (10);
k) WRITE (12);
l) WRITE (16);
m) WRITE AND VERIFY (10);
n) WRITE AND VERIFY (12);
o) WRITE AND VERIFY (16);
p) WRITE SAME (10);
q) WRITE SAME (16);
r) XDWRITEREAD (10); and
s) XDWRITEREAD (32)
t) XPWRITE (10);
u) XPWRITE (32);

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

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

#### 4.22.2.5 Type 3 protection

Type 3 protection:

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

If type 3 protection is enabled, then the following medium commands are invalid and shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

a) READ (32);
b) VERIFY (32);
c) WRITE (32);
d) WRITE AND VERIFY (32); and
e) WRITE SAME (32).

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

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

### 4.22.3 Protection information format

Table 17 defines the placement of protection information in a logical block with a single protection information interval (i.e., the PROTECTION INTERVAL EXPONENT field is set to zero in the parameter list header for a FORMAT UNIT command (see 5.3.2.2)).

**Table 17 — User data and protection information format with a single protection information interval**

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

Table 18 shows an example of the placement of protection information in a logical block with more than one protection information interval (i.e., the PROTECTION INTERVAL EXPONENT field is set to a non-zero value in the

parameter list header for a FORMAT UNIT command (see 5.3.2.2)).

**Table 18 — An example of the user data and protection information format for a logical block with more than one protection information interval**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | USER DATA [first] | | | | | | | |
| ••• | | | | | | | | |
| n - 1 | | | | | | | | |
| n | (MSB) | LOGICAL BLOCK GUARD [first] | | | | | | |
| n + 1 | | | | | | | | (LSB) |
| n + 2 | (MSB) | LOGICAL BLOCK APPLICATION TAG [first] | | | | | | |
| n + 3 | | | | | | | | (LSB) |
| n + 4 | (MSB) | LOGICAL BLOCK REFERENCE TAG [first] | | | | | | |
| ••• | | | | | | | | |
| n + 7 | | | | | | | | (LSB) |
| n + 8 | USER DATA [second] | | | | | | | |
| ••• | | | | | | | | |
| m - 1 | | | | | | | | |
| m | (MSB) | LOGICAL BLOCK GUARD [second] | | | | | | |
| m + 1 | | | | | | | | (LSB) |
| m + 2 | (MSB) | LOGICAL BLOCK APPLICATION TAG [second] | | | | | | |
| m + 3 | | | | | | | | (LSB) |
| m + 4 | (MSB) | LOGICAL BLOCK REFERENCE TAG [second] | | | | | | |
| ••• | | | | | | | | |
| m + 7 | | | | | | | | (LSB) |
| | ⋮ | | | | | | | |
| | USER DATA [last] | | | | | | | |
| ••• | | | | | | | | |
| z - 1 | | | | | | | | |
| z | (MSB) | LOGICAL BLOCK GUARD [last] | | | | | | |
| z + 1 | | | | | | | | (LSB) |
| z + 2 | (MSB) | LOGICAL BLOCK APPLICATION TAG [last] | | | | | | |
| z + 3 | | | | | | | | (LSB) |
| z + 4 | (MSB) | LOGICAL BLOCK REFERENCE TAG [last] | | | | | | |
| ••• | | | | | | | | |
| z + 7 | | | | | | | | (LSB) |

Each USER DATA field shall contain user data.

Each LOGICAL BLOCK GUARD field contains a CRC (see 4.22.4). Only the contents of the USER DATA field immediately preceding THE LOGICAL BLOCK GUARD field. (i.e., the user data between the preceding logical block reference tag, if any, and the current logical block guard) shall be used to generate and check the CRC contained in the LOGICAL BLOCK GUARD field.

Each logical block application tag field is set by the application client. If the device server detects a:

a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.22.2.3) is enabled;
b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 2 protection (see 4.22.2.4) is enabled; or
c) LOGICAL BLOCK APPLICATION TAG field set to FFFFh, logical block reference tag field set to FFFF_FFFFh, and type 3 protection (see 4.22.2.5) is enabled,

then the device server disables checking of all protection information for the associated protection information interval when reading from the medium. Otherwise, if the ATMPE bit in the Control mode page (see SPC-4) is:

a) set to one, then the logical block application tags are defined by the Application Tag mode page (see 6.5.3); or
b) set to zero, then the logical block application tags are not defined by this standard.

The LOGICAL BLOCK APPLICATION TAG field may be modified by a device server if the ato bit is set to zero in the Control mode page (see SPC-4). If the ATO bit is set to one in the Control mode page, then the device server shall not modify the LOGICAL BLOCK APPLICATION TAG field.

The contents of a LOGICAL BLOCK APPLICATION TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

The first lOGICAL BLOCK REFERENCE TAG field of the first logical block in the Data-In Buffer and/or Data-Out Buffer shall contain the value specified in table 19.

**Table 19 — Content of the first LOGICAL BLOCK REFERENCE TAG field for the first logical block in the Data-In Buffer and/or Data-Out Buffer**

| Protection Type | Content of the first LOGICAL BLOCK REFERENCE TAG field for the first logical block in the Data-In Buffer and/or Data-Out Buffer |
|---|---|
| Type 1 [a] protection (see 4.22.2.3) | The least significant four bytes of the LBA contained in the LOGICAL BLOCK ADDRESS field of the command. |
| Type 2 protection (see 4.22.2.4) | The value in the expected initial LOGICAL BLOCK REFERENCE TAG field of the command. |
| Type 3 protection (see 4.22.2.5) | Not defined in this standard. However, this field may be modified by the device server if the ATO bit is set to zero in the Control mode page (see SPC-4). If the ATO bit is set to one in the Control mode page, then the device server shall not modify this field. |

[a] The length of the protection information interval is equal to the logical block length (see 5.3.2).

Subsequent LOGICAL BLOCK REFERENCE TAG fields for a logical block in the Data-In Buffer and/or Data-Out Buffer shall be set as specified in table 20.

**Table 20 — Setting the value in subsequent LOGICAL BLOCK REFERENCE TAG fields for a logical block in the Data-In Buffer and/or Data-Out Buffer**

| Protection Type | The content of subsequent LOGICAL BLOCK REFERENCE TAG fields in the Data-In Buffer and/or Data-Out Buffer |
|---|---|
| Type 1 protection (see 4.22.2.3) and Type 2 protection (see 4.22.2.4) | The previous logical block reference tag plus one. |
| Type 3 protection (see 4.22.2.5) | Not defined in this standard. However, this field may be modified by the device server if the ATO bit is set to zero in the Control mode page (see SPC-4). If the ATO bit is set to one in the Control mode page, then the device server shall not modify this field. |

The contents of a LOGICAL BLOCK REFERENCE TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

### 4.22.4 Logical block guard

#### 4.22.4.1 Logical block guard overview

A LOGICAL BLOCK GUARD field shall contain a CRC that is generated from the contents of only the USER DATA field immediately preceding the LOGICAL BLOCK GUARD field.

Table 21 defines the CRC polynomials used to generate the logical block guard from the contents of the USER DATA field.

**Table 21 — CRC polynomials**

| Function | Definition |
|---|---|
| F(x) | A polynomial representing the transmitted USER DATA field, which is covered by the CRC. For the purposes of the CRC, the coefficient of the highest order term shall be byte zero bit seven of the USER DATA field and the coefficient of the lowest order term shall be bit zero of the last byte of the USER DATA field. |
| F'(x) | A polynomial representing the received USER DATA field. |
| G(x) | The generator polynomial:<br>$G(x) = x^{16} + x^{15} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$<br>(i.e., G(x) = 1_8BB7h) |
| R(x) | The remainder polynomial calculated during CRC generation by the transmitter, representing the transmitted LOGICAL BLOCK GUARD field. |
| R'(x) | A polynomial representing the received LOGICAL BLOCK GUARD field. |
| RB(x) | The remainder polynomial calculated during CRC checking by the receiver.<br>RB(x) = 0 indicates no error was detected. |
| RC(x) | The remainder polynomial calculated during CRC checking by the receiver.<br>RC(x) = 0 indicates no error was detected. |
| QA(x) | The quotient polynomial calculated during CRC generation by the transmitter. The value of QA(x) is not used. |
| QB(x) | The quotient polynomial calculated during CRC checking by the receiver. The value of QB(x) is not used. |
| QC(x) | The quotient polynomial calculated during CRC checking by the receiver. The value of QC(x) is not used. |
| M(x) | A polynomial representing the transmitted USER DATA field followed by the transmitted LOGICAL BLOCK GUARD field. |
| M'(x) | A polynomial representing the received USER DATA field followed by the received LOGICAL BLOCK GUARD field. |

#### 4.22.4.2 CRC generation

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

The transmitter shall calculate the CRC by appending 16 zeros to F(x) and dividing by G(x) to obtain the remainder R(x):

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

R(x) is the CRC value, and is transmitted in the LOGICAL BLOCK GUARD field.

M(x) is the polynomial representing the USER DATA field followed by the LOGICAL BLOCK GUARD field (i.e., F(x) followed by R(x)):

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

#### 4.22.4.3 CRC checking

M'(x) (i.e., the polynomial representing the received USER DATA field followed by the received LOGICAL BLOCK GUARD field) may differ from M(x) (i.e., the polynomial representing the transmitted USER DATA field followed by the transmitted LOGICAL BLOCK GUARD field) if there are transmission errors.

The receiver may check M'(x) validity by appending 16 zeros to F'(x) and dividing by G(x) and comparing the calculated remainder RB(x) to the received CRC value R'(x):

$$\frac{(x^{16} \times F'(x))}{G(x)} = QB(x) + \frac{RB(x)}{G(x)}$$

In the absence of errors in F'(x) and R'(x), the remainder RB(x) is equal to R'(x).

The receiver may check M'(x) validity by dividing M'(x) by G(x) and comparing the calculated remainder RC(x) to zero:

$$\frac{M'(x)}{G(x)} = QC(x) + \frac{RC(x)}{G(x)}$$

In the absence of errors in F'(x) and R'(x), the remainder RC(x) is equal to zero.

Both methods of checking M'(x) validity are mathematically equivalent.

#### 4.22.4.4 CRC test cases

Several CRC test cases are shown in table 22.

**Table 22 — CRC test cases**

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

### 4.22.5 Application of protection information

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

1) determine if a logical unit supports protection information using the INQUIRY command (see the PROTECT bit in the standard INQUIRY data in SPC-4);
2) if protection information is supported, then determine if the logical unit is formatted to accept protection information using the READ CAPACITY (16) command (e.g., see the PROT_EN bit in the returned parameter data (see 5.16.2)); and
3) if the logical unit supports protection information and is not formatted to accept protection information, then format the logical unit with protection information enabled.

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

### 4.22.6 Protection information and commands

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

The commands that are affected when protection information is enabled are listed in table 31 (see 5.1).

Commands that cause a device server to return the length in bytes of each logical block (e.g., the MODE SENSE (10) command and the READ CAPACITY (16) command) shall cause the device server to return the combined length of the USER DATA field(s) contained in the logical block, not including the length of any protection information (i.e., the LOGICAL BLOCK GUARD field(s), the LOGICAL BLOCK APPLICATION TAG field(s), and the LOGICAL BLOCK REFERENCE TAG field(s)) (e.g., if the user data plus the protection information is equal to 520 bytes and there is one protection information interval, then 512 is returned).

## 4.23 Grouping function

A grouping function is a function that collects information about attributes associated with commands (i.e., information about commands with the same group value are collected into the specified group). The definition of the attributes and the groups is outside the scope of this standard. Groups are identified with the GROUP NUMBER field in the CDB of certain commands (e.g., the PRE-FETCH (10) command (see 5.8)).

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

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

Support for the grouping function is indicated in the GROUP_SUP bit in the Extended INQUIRY Data VPD page (see SPC-4).

## 4.24 Background scan operations

### 4.24.1 Background scan overview

A background scan operation is either a background pre-scan operation (see 4.24.2) or a background medium scan operation (see 4.24.3).

During a background scan operation, the device server performs read operations for the purpose of:

a) identifying logical blocks that are difficult to read (i.e., recoverable) or unreadable (i.e., unrecoverable);
b) logging problems encountered during the background scan operation; and
c) when allowed, taking a vendor specific action to repair recoverable logical blocks or perform automatic read reallocation of recoverable logical blocks.

If, during a background scan operation, a read operation encounters a recovered error (i.e., a logical block is readable but requires extra actions (e.g., retries or application of a correction algorithm) to be read), then the device server may resolve the problem using vendor specific means. The value of the ARRE bit in the Read-Write Error Recovery mode page (see 6.5.8) determines whether or not the device server performs automatic read reassignment.

If, during a background scan operation, a read operation encounters an unrecovered error (i.e., a logical block is unreadable), then the device server may mark the logical block unrecoverable. The value of the AWRE bit in the Read-Write Error Recovery mode page (see 6.5.8) determines whether or not the device server performs automatic write reassignment. If allowed by the AWRE bit setting, the device server performs automatic write reassignment at the start of the next write medium operation to that logical block.

During a background scan operation, the device server:

a) may scan the logical blocks in any order (e.g., based on physical block layout);
b) should not retain any data from logical blocks in cache memory after the logical blocks are read;
c) shall ignore pseudo unrecovered errors with correction disabled (see 4.18.2); and
d) shall process pseudo unrecovered errors with correction enabled.

### 4.24.2 Background pre-scan operations

#### 4.24.2.1 Enabling background pre-scan operations

A background pre-scan operation is enabled after:

1) the EN_PS bit in the Background Control mode page (see 6.5.4) is set to zero:
2) the EN_PS bit is set to one; and
3) the SCSI device is power cycled if;
   A) the S_L_FULL bit in the Background Control mode page is:
      a) set to zero; or
      b) set to one and the Background Scan log parameters in the Background Scan Results log page (see 6.4.2) are not all used;

      and
   B) the saved value of the EN_PS bit is set to one.

After a background pre-scan operation is enabled, the device server shall:

a) initialize the Background Pre-scan Time Limit timer to the time specified in the PRE-SCAN TIME LIMIT field in the Background Control mode page and start the timer;
b) initialize the Background Medium Scan Interval timer to the time specified in the BACKGROUND MEDIUM SCAN INTERVAL TIME field in the Background Control mode page and start the timer; and
c) begin the background pre-scan operation (i.e., begin scanning the medium).

#### 4.24.2.2 Suspending and resuming background pre-scan operations

A background pre-scan operation shall be suspended when any of the following occurs:

a) a command or task management function is processed that requires the background pre-scan operation to be suspended;
b) a SCSI event (e.g., a hard reset) (see SAM-5) is processed that requires the background pre-scan operation to be suspended;
c) a power condition timer expires (see the Power Condition mode page in SPC-4), and the PM_BG_PRECEDENCE field in the Power Condition mode page is set to 10b; or
d) the S_L_FULL bit in the Background Control mode page (see 6.5.4) is set to one, and the Background Scan log parameters (see 6.4.2) are all used.

When a command is received that requires a background pre-scan operation to be suspended, the following should occur within the time specified in the MAXIMUM TIME TO SUSPEND BACKGROUND SCAN field in the Background Control mode page:

a) the logical unit suspends the background medium scan operation; and
b) the device server begins processing the command.

When a background pre-scan operation is suspended, the device server shall not stop:

a) the Background Pre-scan Time Limit timer;
b) the Background Medium Scan Interval timer; or
c) any process that results in an event that causes a background function to occur (e.g., not stop any timers or counters associated with background functions).

While a background pre-scan operation is suspended and not halted (see 4.24.3.2), the device server shall convert each write medium operation accessing a logical block that has not been scanned during the background pre-scan operation into a write medium operation followed by a verify medium operation in order to verify that the data just written was read back without error. If a write medium operation accesses a logical

block that has already been scanned during the background pre-scan operation, then the device server shall process the write medium operation without the additional verify medium operation.

A background pre-scan operation shall be resumed from where the operation was suspended when:

a) there are no commands in any task set to be processed;
b) there are no task management functions to be processed;
c) there are no SCSI events to be processed;
d) no ACA condition exists;
e) the PM_BG_PRECEDENCE field in the Power Condition mode page is set to 10b (see SPC-4), but no power condition timer defined in the Power Condition mode page has expired;
f) the S_L_FULL bit in the Background Control mode page is set to zero, or the Background Medium Scan log parameters in the Background Scan Results log page are not all used;
g) the logical unit has been idle for the time specified in the MINIMUM IDLE TIME BEFORE BACKGROUND SCAN field in the Background Control mode page; and
h) the background pre-scan operation has not been halted (see 4.24.3.2).

#### 4.24.2.3 Halting background pre-scan operations

The device server shall halt a background pre-scan operation if any of the following occurs:

a) the background pre-scan operation completes scanning all logical blocks on the medium;
b) an application client sets the EN_PS bit to zero in the Background Control mode page (see 6.5.4);
c) the Background Pre-scan Time Limit timer expires;
d) the device server detects a fatal error;
e) the device server detects a vendor specific pattern of errors;
f) the device server detects a medium formatted without a PLIST (see 4.13); or
g) the device server detects temperature out of range.

After a background pre-scan operation has been halted, the device server shall use the procedure to enable a background pre-scan operation to begin a new background pre-scan operation (see 4.24.2.1).

### 4.24.3 Background medium scan

#### 4.24.3.1 Enabling background medium scan operations

Background medium scan operations are enabled if:

a) a background pre-scan operation (see 4.24.2) is not in progress;
b) the S_L_FULL bit in the Background Control mode page (see 6.5.4) is:
   A) set to zero; or
   B) set to one and the Background Scan log parameters in the Background Scan Results log page (see 6.4.2) are not all used;
   and
c) the EN_BMS bit in the Background Control mode page is set to one.

If background medium scan operations are enabled, then the device server shall begin a background medium scan operation (i.e., begin scanning the medium) when:

a) the Background Medium Scan Interval timer has expired; and
b) the logical unit has been idle for the time specified in the MINIMUM IDLE TIME BEFORE BACKGROUND SCAN field in the Background Control mode page.

After power on, if background pre-scan operations are not enabled (see 4.24.2.1), then the device server shall set the Background Medium Scan Interval timer to zero (i.e., expired).

Whenever a background medium scan operation begins, the device server shall set the Background Medium Scan Interval timer to the time specified in the BACKGROUND MEDIUM SCAN INTERVAL TIME field in the Background Control mode page and start the timer.

#### 4.24.3.2 Suspending and resuming background medium scan operations

The logical unit shall suspend a background medium scan operation if any of the following occurs:

a) a command or task management function is processed that requires the background medium scan operation to be suspended;
b) a SCSI event (e.g., a hard reset) (see SAM-5) is processed that requires the background medium scan operation to be suspended;
c) a power condition timer expires (see the Power Condition mode page in SPC-4), and the PM_BG_PRECEDENCE field in the Power Condition mode page is set to 10b;
d) the S_L_FULL bit in the Background Control mode page (see 6.5.4) is set to one, and the Background Scan log parameters (see 6.4.2) are all used; or
e) an application client sets the EN_BMS bit in the Background Control mode page to zero.

When a command is received that requires a background medium scan operation to be suspended, the following should occur within the time specified in the MAXIMUM TIME TO SUSPEND BACKGROUND SCAN field in the Background Control mode page:

a) the logical unit suspends the background medium scan operation; and
b) and the device server begins processing the command.

When a background pre-scan operation is suspended, the device server shall not stop:

a) the Background Medium Scan Interval timer; or
b) any process that results in an event that causes a background function to occur (e.g., not stop any timers or counters associated with background functions).

The logical unit shall resume a suspended background medium scan operation from where the operation was suspended when:

a) there are no commands in any task set to be processed;
b) there are no task management functions to be processed;
c) there are no SCSI events to be processed;
d) no ACA condition exists;
e) the PM_BG_PRECEDENCE field in the Power Condition mode page is set to 10b (see SPC-4), but no power condition timer defined in the Power Condition mode page has expired;
f) the S_L_FULL bit in the Background Control mode page is set to zero, or the Background Medium Scan log parameters in the Background Scan Results log page are not all used;
g) the EN_BMS bit in the Background Control mode page is set to one; and
h) the logical unit has been idle for the time specified in the MINIMUM IDLE TIME BEFORE BACKGROUND SCAN field in the Background Control mode page.

#### 4.24.3.3 Halting background medium scan operations

The device server shall halt background medium scan operations if any of the following occurs:

a) the background medium scan operation completes scanning all logical blocks on the medium;
b) the device server detects a fatal error;
c) the device server detects a vendor specific pattern of errors;
d) the device server detects a medium formatted without a PLIST (see 4.13); or
e) the device server detects temperature out of range.

After background medium scan operations have been halted, the device server shall use the procedure to enable background medium scan operations (see 4.24.3.1) to re-enable the background medium scan function.

### 4.24.4 Interpreting the logged background scan results

An application client may:

a) poll the Background Scan Results log page (see 6.4.2) to get information about background pre-scan and background medium scan activity; or
b) use the EBACKERR bit and the MRIE field in the Informational Exceptions Control mode page (see 6.5.6) to select a method of indicating that a medium error was detected.

If the EBACKERR bit is set to one and a medium error was detected, then the device server shall return the following additional sense codes using the method defined by the value in the MRIE field:

a) WARNING - BACKGROUND PRE-SCAN DETECTED MEDIUM ERROR if the failure occurs during a background pre-scan operation; or
b) WARNING - BACKGROUND MEDIUM SCAN DETECTED MEDIUM ERROR if the failure occurs during a background medium scan operation.

The Background Scan Status log parameter (see 6.4.2.2) in the Background Scan Results log page (see 6.4.2) indicates:

a) whether or not a background scan operation is active or halted;
b) the number of background scan operations that have been performed on the medium; and
c) the progress of a background scan operation that is active.

This information may be used by an application client to monitor the background scan operations and should be used by an application client after notification via an informational exception (see 6.5.6).

The Background Scan Results log parameters (see 6.4.2.3), if any, in the Background Scan Results log page describe the LBA and the reassignment status of each logical block that generated recovered errors or unrecovered errors during the background scan's read operations.

After an application client analyzes the Background Scan Results log parameters and has completed actions, if any, to repair any of the indicated LBAs, the application client may delete all Background Scan Results parameters by issuing a LOG SELECT command (e.g., with the PCR bit set to one in the CDB, or with the PC field set to 11b and the PARAMETER LIST LENGTH field set to zero in the CDB) (see SPC-4).

A background medium scan operation may continue to run during log page accesses. To ensure that the values in the Background Scan Results log page do not change during a sequence of accesses, the application client:

1) sets the EN_BMS bit to zero in the Background Control mode page in order to suspend the background medium scan operation;
2) reads the Background Scan Results log page with a LOG SENSE command;
3) processes the Background Scan Results log page;
4) deletes the Background Scan Results log page entries with the LOG SELECT command (e.g., with the PCR bit set to one in the CDB); and
5) sets the EN_BMS bit to one in the Background Control mode page in order to re-enable the background scan operation.

### 4.25 Association between commands and CbCS permission bits

Table 23 defines the Capability based Command Security (i.e., CbCS) permissions required for each command defined in this standard. The permissions shown in table 23 are defined in the PERMISSIONS BIT MASK field in the CbCS capability descriptor in a CbCS extension descriptor (see SPC-4). This standard does not define any permissions specific to block commands.

**Table 23 — Associations between commands and CbCS permissions** (part 1 of 2)

| Command | Permissions bit mask bits [a] | | | | |
|---|---|---|---|---|---|
| | DATA READ | DATA WRITE | PARM READ | PARM WRITE | PHY ACC |
| COMPARE AND WRITE | | 1 | | | |
| FORMAT UNIT | | 1 | | 1 | |
| GET LBA STATUS | | | 1 | | |
| ORWRITE (16) / (32) | | 1 | | | |
| POPULATE TOKEN | 1 | | | | |
| PRE-FETCH (10) / (16) | 1 | | | | |
| PREVENT ALLOW MEDIUM REMOVAL | | | | | 1 |
| READ (10) / (12) / (16) / (32) | 1 | | | | |
| READ CAPACITY (10) / (16) | | | 1 | | |
| READ DEFECT DATA (10) / (12) | | | 1 | | |
| READ LONG (10) / (16) | 1 | | | | |
| REASSIGN BLOCKS | | | | | 1 |
| RECEIVE ROD TOKEN INFORMATION | See SPC-4 | | | | |
| REPORT REFERRALS | | | 1 | | |
| START STOP UNIT | | | | | 1 |
| SYNCHRONIZE CACHE (10) / (16) | | 1 | | | |
| UNMAP | | 1 | | | |
| VERIFY (10) / (12) / (16) / (32) | 1 | | | | |
| WRITE (10) / (12) / (16) / (32) | | 1 | | | |
| WRITE AND VERIFY (10) / (12) / (16) / (32) | | 1 | | | |
| WRITE LONG (10) / (16) | | 1 | | | |

[a] A device server shall only process a command shown in this table as specified by the CDB field of an extended CDB (see SPC-4) that contains a CbCS capability descriptor when all of the bits marked with a "1" in the row for that command are set to one in the PERMISSIONS BIT MASK field in that descriptor. The permissions bits represented by the empty cells in a row are ignored. If a device server receives a command specified by the CDB field of an extended CDB that does not contain the CbCS capability descriptor with all of the bits set to one as defined in this table, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

**Table 23 — Associations between commands and CbCS permissions** (part 2 of 2)

| Command | Permissions bit mask bits [a] | | | | |
|---|---|---|---|---|---|
| | DATA READ | DATA WRITE | PARM READ | PARM WRITE | PHY ACC |
| WRITE SAME (10) / (16) / (32) | | 1 | | | |
| WRITE USING TOKEN | | 1 | | | |
| XDWRITEREAD (10) / (32) | 1 | 1 | | | |
| XPWRITE (10) / (32) | | 1 | | | |

[a] A device server shall only process a command shown in this table as specified by the CDB field of an extended CDB (see SPC-4) that contains a CbCS capability descriptor when all of the bits marked with a "1" in the row for that command are set to one in the PERMISSIONS BIT MASK field in that descriptor. The permissions bits represented by the empty cells in a row are ignored. If a device server receives a command specified by the CDB field of an extended CDB that does not contain the CbCS capability descriptor with all of the bits set to one as defined in this table, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

### 4.26 Deferred microcode activation

After receiving a FORMAT UNIT command (see 5.3) or a START STOP UNIT command (see 5.25), a device server shall, prior to processing the command, activate any deferred microcode that has been downloaded as a result of a WRITE BUFFER command with the mode field set to 0Eh (see SPC-4), prior to processing the command.

### 4.27 Model for uninterrupted sequences on LBA ranges

Direct-access block devices may perform commands that require an uninterrupted series of actions to be performed on a specified range of LBAs. The uninterrupted sequences are relative to the consistency of the data in the logical blocks specified by the particular command that requires the uninterrupted sequence. The uninterrupted sequences do not impact the processing of commands that access logical blocks other than those specified in the command requiring an uninterrupted sequence. The task attribute (see SAM-5) controls interactions between multiple commands. Commands with uninterrupted sequences on LBA ranges are:

a) COMPARE AND WRITE;
b) ORWRITE (16);
c) ORWRITE (32);
d) XDWRITEREAD (10);
e) XDWRITEREAD (32);
f) XPWRITE (10); and
g) XPWRITE (32).

### 4.28 Referrals

#### 4.28.1 Referrals overview

Referrals allow a logical unit to inform an application client that one or more user data segments (i.e., ranges of logical blocks) are accessible through target port group(s).

Support for referrals is indicated by the device server setting the R_SUP bit to one in the Extended INQUIRY Data VPD page (see SPC-4).

An application client may determine information on referrals by:

a) issuing commands; or
b) monitoring sense data returned as part of a completed command or a terminated command.

Figure 11 shows an example of how a logical unit informs an application client that one or more user data segments are accessible through target port groups.

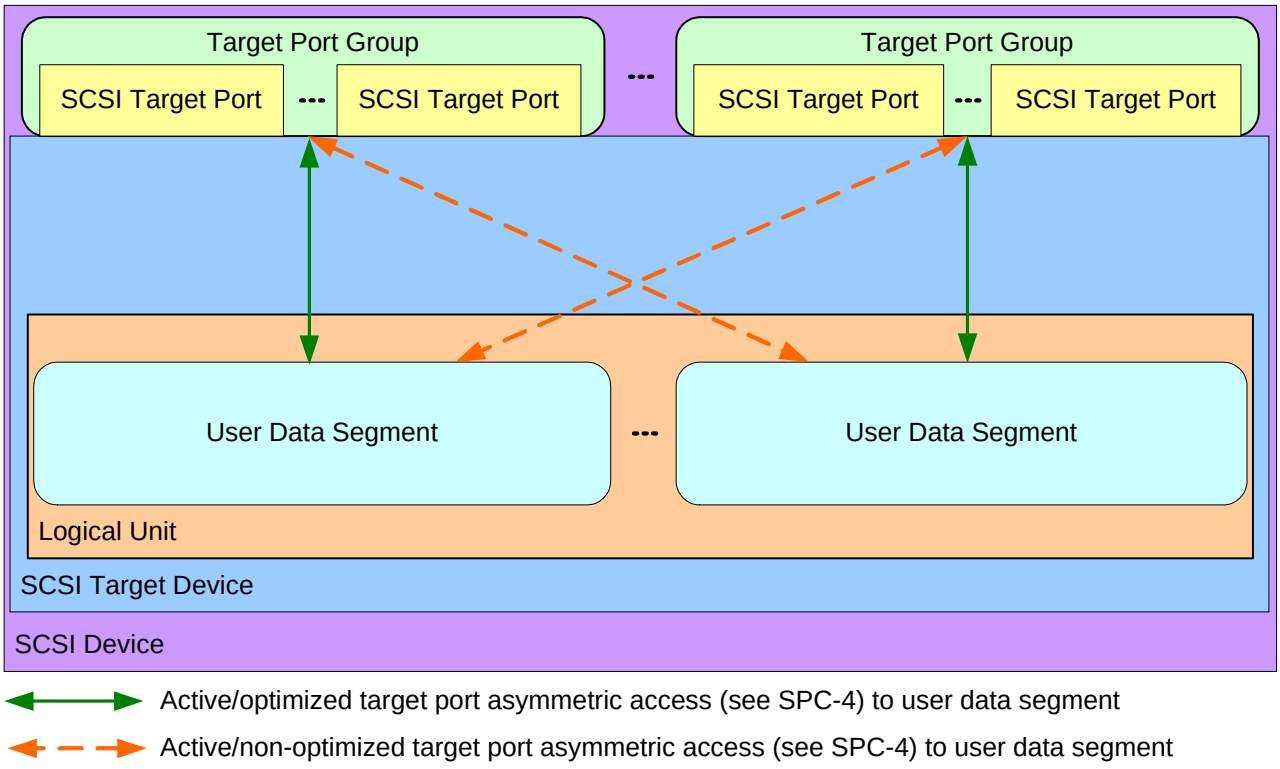



**Figure 11 — Referrals**

### 4.28.2 Discovering referrals

An application client may determine referrals information on a logical unit by:

1) determining if the R_SUP bit is set to one in the Extended INQUIRY Data VPD page (i.e., logical unit supports referrals) (see SPC-4);
2) requesting the user data segment information from the Referrals VPD page (see 6.6.5);
3) requesting a list of target port groups by issuing a REPORT TARGET PORT GROUPS command (see SPC-4); and
4) either:
   A) requesting referrals information by issuing a REPORT REFERRALS command (see 5.23); or
   B) monitoring for referral information in sense data returned by the device server (see 4.28.4).

The following calculation is used to determine the first LBA for each user data segment within the range of LBAs indicated by the user data segment referral descriptors (see table 86) returned in response to a REPORT REFERRALS command or the user data segment referrals sense data descriptor (see 4.18.4):

first LBA of the current user data segment = first LBA + (segment size × segment multiplier)

where:

| | |
|---|---|
| first LBA | the initial value is the first user data segment LBA specified in the user data segment referral descriptor (see table 14). Subsequent values, if any, are the first LBA of the previous user data segment; |
| segment size | the content of the USER DATA SEGMENT SIZE field (see 6.6.5); and |
| segment multiplier | the content of the USER DATA SEGMENT MULTIPLIER field (see 6.6.5). |

If the content of the USER DATA SEGMENT SIZE field is greater than zero, and the content of the USER DATA SEGMENT MULTIPLIER field is greater than zero, then the following calculation may be used to determine the last

LBA for each user data segment within the range of LBAs indicated by the user data segment referral descriptors (see table 86) returned in response to a REPORT REFERRALS command or the user data segment referrals sense data descriptor (see 4.18.4):

last LBA of the current user data segment = first LBA + (segment size – 1)

where:

first LBA the first LBA of the current user data segment;

segment size the content of the USER DATA SEGMENT SIZE field (see 6.6.5).

If the content of the USER DATA SEGMENT SIZE field is zero, then there is only one user data segment, and the last LBA of that user data segment is equal to the last LBA specified in the last user data segment lba field (see table 14).

### 4.28.3 Discovering referrals example

#### 4.28.3.1 Referrals example with no user data segment multiplier

This subclause demonstrates a method an application client my use to determine the optimal target port group from which to access logical blocks using information sent from the device server when the user data segment multiplier is set to zero.

Figure 12 shows an example of a SCSI device in which referrals have been implemented with a user data segment multiplier of zero.

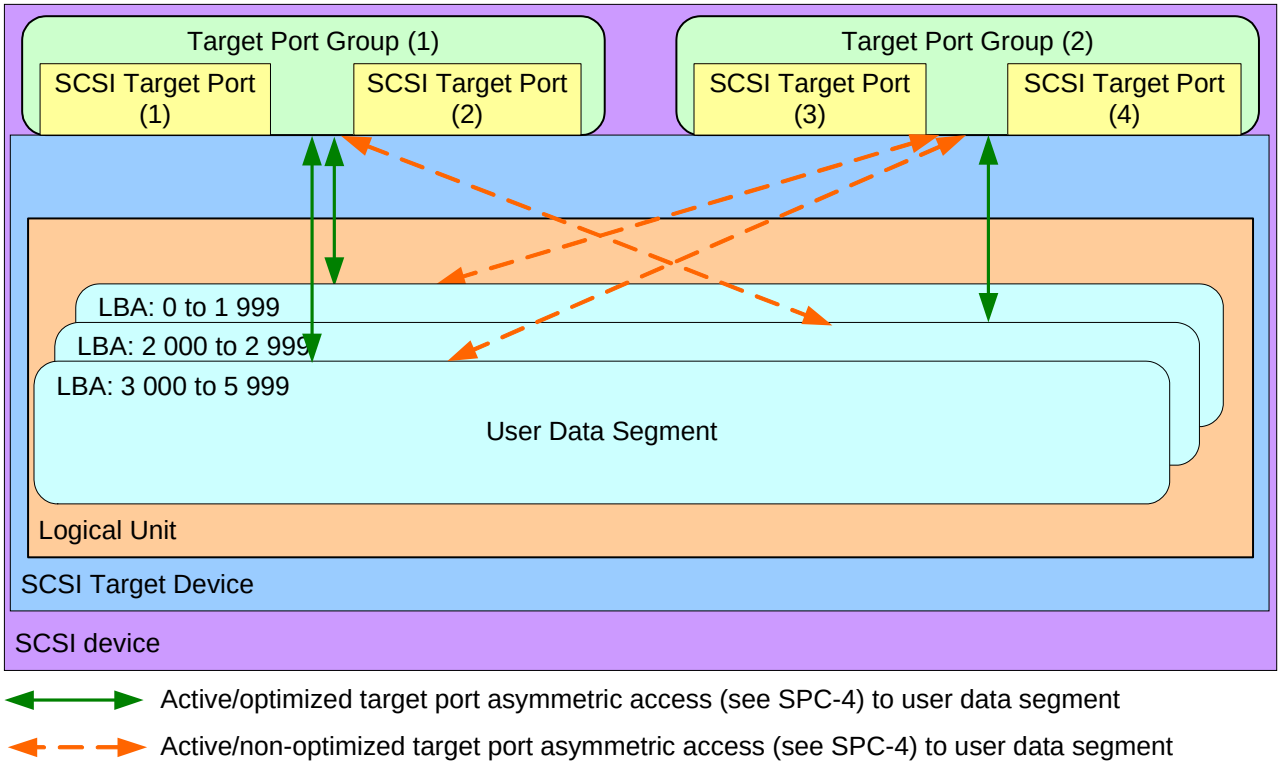



**Figure 12 — Referrals example with no user data segment multiplier**

In the example shown in figure 12, the application client acquires the information from the logical unit as shown in table 24.

**Table 24 — Example of referrals application client information with no user data segment multiplier**

| **INQUIRY command Referrals VPD page** | | | |
|---|---|---|---|
| **User data segment size** | | **User data segment multiplier** | |
| ignored | | 0 | |
| **REPORT TARGET PORT GROUPS command** | | | |
| **Asymmetric access state** | **Target port group** | **Relative target port identifier** | |
| 4h (i.e., logical block dependent) | 1 | 1 | |
| | | 2 | |
| | 2 | 3 | |
| | | 4 | |
| **REPORT REFERRALS command or user data segment referral sense data descriptors** | | | |
| **First user data segment LBA** | **Last user data segment LBA** | **Asymmetric access state** | **Target port group** |
| 0 | 1 999 | 0 (i.e., active/optimized) | 1 |
| | | 1 (i.e., active/non-optimized) | 2 |
| 2 000 | 2 999 | 0 (i.e., active/optimized) | 2 |
| | | 1 (i.e., active/non-optimized) | 1 |
| 3 000 | 5 999 | 0 (i.e., active/optimized) | 1 |
| | | 1 (i.e., active/non-optimized) | 2 |

The application may determine the user data segments that are optimally accessed through the two target port groups as shown in table 25.

**Table 25 — User data segment calculations with no user data segment multiplier**

| **First LBA of user data segment** | **Calculation (see 4.28.2)** | **Last LBA of user data segment** | **Calculation (see 4.28.2)** |
|---|---|---|---|
| **Target port group 1 user data segments in active/optimized asymmetric access state** | | | |
| 0 | 0 [a] | 1 999 | 1 999 [b] |
| 3 000 | 3 000 [a] | 5 999 | 5 999 [b] |
| **Target port group 2 user data segments in active/optimized asymmetric access state** | | | |
| 2 000 | 2 000 [a] | 2 999 | 2 999 [b] |

[a] The first user data segment LBA
[b] The last user data segment LBA

#### 4.28.3.2 Referrals example with non-zero user data segment multiplier

This subclause demonstrates a method that an application client my use to determine the optimal target port group from which to access logical blocks using information sent from the device server when the user data segment multiplier is set to a non-zero value.

Figure 13 shows an example of a SCSI device in which referrals have been implemented with a user data segment multiplier of two and a user data segment size of 1 000.

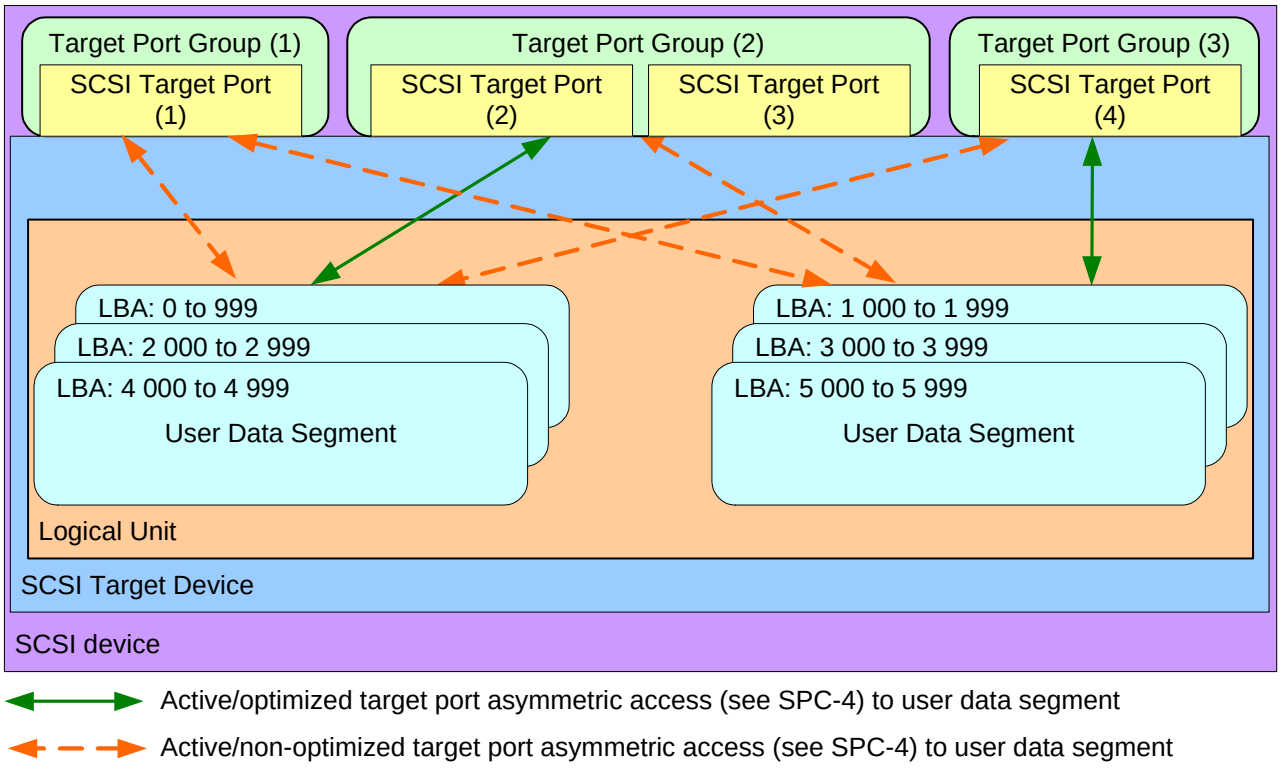



**Figure 13 — Referrals example with non-zero user data segment multiplier**

In the example shown in figure 13, the application client acquires the information from the logical unit as shown in table 26.

**Table 26 — Example of referrals application client information with non-zero user data segment multiplier**

| INQUIRY command Referrals VPD page | | | |
|---|---|---|---|
| **User data segment size** | | **User data segment multiplier** | |
| 1 000 | | 2 | |
| **REPORT TARGET PORT GROUPS command** | | | |
| **Asymmetric access state** | **Target port group** | **Relative target port identifier** | |
| 4h (i.e., logical block dependent) | 1 | 1 | |
| | 2 | 2 | |
| | | 3 | |
| | 3 | 4 | |
| **REPORT REFERRALS command or user data segment referral sense data descriptors** | | | |
| **First user data segment LBA** | **Last user data segment LBA** | **Asymmetric access state** | **Target port group** |
| 0 | 4 999 | 0 (i.e., active/optimized) | 2 |
| | | 1 (i.e., active/non-optimized) | 1 |
| | | 1 (i.e., active/non-optimized) | 3 |
| 1 000 | 5 999 | 0 (i.e., active/optimized) | 3 |
| | | 1 (i.e., active/non-optimized) | 1 |
| | | 1 (i.e., active/non-optimized) | 2 |

The application may determine the user data segments that are optimally accessed through the two target port groups as shown in table 27.

**Table 27 — User data segment calculations with non-zero user data segment multiplier**

| First LBA of user data segment | Calculation (see 4.28.2) | Last LBA of user data segment | Calculation (see 4.28.2) |
|---|---|---|---|
| **Target port group 2 user data segments in active/optimized asymmetric access state** | | | |
| 0 | 0 [a] | 999 | 0 [a] + (1 000 – 1) |
| 2 000 | 0 + (1 000 × 2) | 2 999 | 2 000 + (1 000 – 1) |
| 4 000 | 2 000 + (1 000 × 2) | 4 999 | 4 000 + (1 000 – 1) |
| **Target port group 3 user data segments in active/optimized asymmetric access state** | | | |
| 1 000 | 1 000 [a] | 1 999 | 1 000 [a] + (1 000 – 1) |
| 3 000 | 1 000 + (1 000 × 2) | 3 999 | 3 000 + (1 000 – 1) |
| 5 000 | 3 000 + (1 000 × 2) | 5 999 | 5 000 + (1 000 – 1) |

[a] The first user data segment LBA.

### 4.28.4 Referrals in sense data

Returning referral information in sense data is enabled if the:

a) R_SUP bit is set to one in the Extended INQUIRY Data VPD page (i.e., logical unit supports referrals) (see SPC-4); and
b) D_SENSE bit in the Control mode page is set to one (i.e., returning descriptor formatted sense data is enabled) (see SPC-4).

If reporting of referrals in sense data is enabled, a command completes without error, no other sense data is available within the logical unit, and the device server has an alternate I_T_L nexus that an application client should use to access at least one of the specified logical blocks, then the device server shall complete the command with GOOD status with the sense key set to COMPLETED, the additional sense code set to INSPECT REFERRALS SENSE DESCRIPTORS, and a user data segment referrals sense data descriptor (see 4.18.4).

The user data segment referral sense data descriptor (see 4.18.4) shall define the description of as many complete user data segments (i.e., one user data segment referral descriptor contains one complete user data segment) that fit in the maximum number of bytes allowed for sense data (i.e., 244 bytes). If all the user data segments do not fit within the maximum number of bytes allowed for sense data, then:

a) the device server shall set the NOT_ALL_R bit to one in the user data segment referral sense data descriptor (see 4.18.4); and
b) the selection of which user data segments to include in the user data segment referral sense data descriptor is vendor specific.

Each user data segment referral sense data descriptor (see 4.18.4) contains information on alternate I_T_L nexuses to user data segments that the application client should use to access logical blocks within the LBA range(s) indicated by the user data segments.

If reporting of referrals in sense data is enabled, the device server receives a command for which the device server is not able to access user data associated with the requested command, and the inaccessible user data is accessible through another target port group, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ABORTED COMMAND, the additional sense code set to INSPECT REFERRALS SENSE DESCRIPTORS, and a user data segment referral sense data descriptor. The user data segment referral sense data descriptor shall, at a minimum, indicate the user data segment that contains the address of the first inaccessible logical block. Any other type of error that occurs while processing the command shall take precedence and be reported as described in this standard. If any other type of error that occurs while the device server is processing the command, then processing that error shall take precedence, over processing this command, and the device server shall report the error as described in this standard.

If reporting of in sense data referrals is disabled (see 4.28.1), the device server receives a command for which the device server is not able to access user data associated with the requested command, and the inaccessible user data is accessible through another target port group, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to HARDWARE ERROR and the additional sense code set to INTERNAL TARGET FAILURE.

## 4.29 ORWRITE commands

### 4.29.1 ORWRITE commands overview

The ORWRITE commands (see 5.5 and 5.6) provide a mechanism for an application client to manipulate the bitmap structures on direct-access block devices.

The ORWRITE commands shall be processed by the device server performing the following as an uninterrupted sequence of actions (see 4.27):

1) if required by the rules for caching (see 4.15), then perform read medium operations on the logical block(s) referenced by the LBAs specified by this command;
2) transfer the specified number of logical blocks from the Data-Out Buffer;
3) perform the specified Boolean arithmetic function on:
   A) the user data contained in the logical blocks read from the medium or cache; and
   B) the user data contained in the logical blocks transferred from the Data-Out Buffer;
4) store the results of the function in a bitmap buffer (see 3.2.9);
5) generate new protection information, if any, from the stored results;
6) store the generated protection information, if any, into the bitmap buffer; and
7) write the updated logical block data from the bitmap buffer based on the rules for caching.

If the read of the protection information from the medium is successful, and the check of the protection information transferred from the Data-Out Buffer is successful, then the device server shall generate the new protection information (see 4.22) as follows:

a) set the logical block guard field to the CRC (see 4.22.4) generated from the bitmap buffer by the device server;
b) set the logical block reference tag field to the logical block reference tag field received from the Data-Out Buffer; and
c) set the logical block application tag field to the logical block application tag field received from the Data-Out Buffer.

In order to support the manipulation of bitmap structures:

a) the ORWRITE (16) command supports the set operation (see 4.29.4); and
b) the ORWRITE (32) command supports:
   A) the set operation; and
   B) the change generation and clear operation (see 4.29.3).

### 4.29.2 ORWgeneration code

#### 4.29.2.1 ORWgeneration code overview

The ORWRITE commands use a generation code for synchronization. The device server shall establish and maintain the following generation codes:

a) a current ORWgeneration code; and
b) a previous ORWgeneration code.

Subsequent ORWRITE command processing by the device server is dependent on comparisons involving the ORWgeneration codes. Changes in these ORWgeneration codes define a synchronization point in the management of the bitmap.

#### 4.29.2.2 ORWgeneration code processing

The device server shall maintain at least one current ORWRITE processing policy. The device server may support more than one ORWRITE processing policy (see table 28 in 4.29.4).

When processing an ORWRITE (32) command (see 5.6), the device server compares the value in the EXPECTED ORWGENERATION field in the CDB to the current ORWgeneration code (see 4.29.2.1), and:

a) if the two values are equal, then the device server continues processing the ORWRITE (32) command; or

b) if the two values are not equal, then, for a set operation, the current ORWRITE processing policy (see table 28) determines how the device server continues processing the ORWRITE (32) command.

If the device server supports both the ORWRITE (16) command (see 5.5) and the ORWRITE (32) command, then the device server shall process all ORWRITE (16) commands as if they contained an EXPECTED ORWGENERATION field set to zero. If the current ORWgeneration code is non-zero, then the current ORWRITE processing policy (see table 28) determines how the device server shall process the command.

### 4.29.3 Change generation and clear operation

The change generation portion of the change generation and clear operation is used to establish a point of synchronization. The clear portion of the change generation and clear operation is used to set zero or more bits in the bitmap structure to zero.

The device server processes a change generation and clear operation if:

a) the BMOP field in the ORWRITE (32) (see 5.6) command is set to 001b; and
b) the value in the EXPECTED ORWGENERATION field is equal to the current ORWgeneration code in the device server.

If the device server processes a change generation and clear operation, then the device server shall perform the following as an uninterrupted sequence:

1) read the specified logical blocks (i.e., user data (see 3.2.90) and protection information (see 3.2.61), if any) from the medium;
2) transfer the specified logical blocks from the Data-Out Buffer (see 3.2.19);
3) perform an AND operation (see 3.2.3) on the user data contained in the logical blocks read from the medium and the user data contained in the logical blocks transferred from the Data-Out Buffer;
4) store the result of the operation in a bitmap buffer (see 3.2.9);
5) generate new protection information, if any, from the stored results;
6) store the generated protection information, if any, into the bitmap buffer;
7) write the updated logical blocks (i.e., those containing the user data resulting from the AND operation and the generated protection information, if any) from the bitmap buffer to the medium;
8) set the current ORWRITE processing policy to the value in the PREVIOUS GENERATION PROCESSING field in the ORWRITE (32) command;
9) set the previous ORWGeneration code (see 4.29.2) to the current ORWgeneration code in the device server; and
10) set the current ORWGeneration code (see 4.29.2) to the value in the NEW ORWGENERATION field in the ORWRITE (32) command.

If the value in the EXPECTED ORWGENERATION field is not equal to the current ORWgeneration code, then the device server shall terminate the ORWRITE (32) command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to ORWRITE GENERATION DOES NOT MATCH.

If a power on or hard reset condition occurs, then the device server shall set:

a) the current ORWGeneration code to zero;
b) the previous ORWGeneration code to zero; and
c) the current ORWRITE processing policy to 7h.

The device server shall preserve the following across a logical unit reset:

a) the current ORWGeneration code;
b) the previous ORWGeneration code; and
c) the current ORWRITE processing policy.

### 4.29.4 Set operation

The set operation is used to set zero or more bits in the bitmap structure to one.

The device server processes a set operation for an ORWRITE command (see 5.5 and 5.6) if the BMOP field in the CDB is set to 000b.

The device server shall perform a set operation by performing the actions specified in table 28, which shows the current ORWgeneration code, the previous ORWgeneration code, and the device server's current ORWRITE processing policy (see 4.29.3).

**Table 28 — ORWRITE set processing**

| Current ORWRITE processing policy | Description | The value in the expected ORWGENERATION field matches | | |
|---|---|---|---|---|
| | | Current ORWgeneration code | Previous ORWgeneration code | Any other value |
| 0h | Previous generation writes to the medium | PA | PA | CCG |
| 1h | Reserved | | | |
| 2h | Previous generation does not change the medium | PA | DN | CCG |
| 3h | Non-current generation writes to the medium | PA | PA | PA |
| 4h | Reserved | | | |
| 5h | Non-current generation does not change the medium | PA | DN | DN |
| 6h | Reserved | | | |
| 7h | Non-current generation codes are not processed | PA | CCG | CCG |
| 8h to Fh | Reserved | | | |

Key:

PA = the device server shall perform the following as an uninterrupted sequence:

1) read the specified logical block data from the medium;
2) transfer the specified logical blocks from the Data-Out Buffer (see 3.2.19);
3) perform an OR operation (see 3.2.53) on the logical blocks read from the medium and the user data contained in the logical blocks transferred from the Data-Out Buffer;
4) store the results in a bitmap buffer (see 3.2.9);
5) generate new protection information, if any, from the stored results;
6) store the generated protection information, if any, into the bitmap buffer; and
7) write the updated logical blocks (i.e., those containing the user data resulting from the OR operation and the generated protection information, if any) from the bitmap buffer to the medium;

DN = the device server shall discard the contents of the Data-Out Buffer and shall complete the command with GOOD status.

CCG = the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to ORWRITE GENERATION DOES NOT MATCH

## 4.30 Block device ROD token operations

### 4.30.1 Block device ROD token operations overview

Application clients request that block device ROD token operations (see SPC-4) be performed using the commands summarized in this subclause.

Copy managers (see SPC-4) that implement the POPULATE TOKEN command (see 5.7) or the WRITE USING TOKEN command (see 5.46) shall implement the following:

a) the POPULATE TOKEN command;
b) the WRITE USING TOKEN command;
c) the RECEIVE COPY STATUS (LID4) command (see SPC-4);
d) the RECEIVE ROD TOKEN INFORMATION command (see SPC-4 and 5.22); and
e) the Third-party Copy VPD page (see SPC-4) containing at least one Block Device ROD Token Limits descriptor (see 6.6.6.3).

The POPULATE TOKEN command may cause the copy manager to create zero or one point in time ROD tokens. If the POPULATE TOKEN command causes one point in time ROD token to be created, then this point in time ROD token may be retrieved by an application client using the RECEIVE ROD TOKEN INFORMATION command.

The WRITE USING TOKEN command causes the copy manager to transfer the data represented by the specified ROD token (i.e., the data represented by the ROD token retrieved using the RECEIVE ROD TOKEN INFORMATION command or the data represented by the block device zero ROD token).

The copy manager manages the point in time ROD token.

After the copy manager begins processing a POPULATE TOKEN command or a WRITE USING TOKEN command, the copy manager shall preserve information for return in response to a RECEIVE ROD TOKEN INFORMATION command as defined in SPC-4.

Block device range descriptor lists (see 5.7.3) contain non-overlapping block device range descriptors and are used by the application client to specify:

a) the logical blocks to include in the data represented by the ROD token;
b) the sequence of the logical blocks in the data represented by the ROD token (e.g., the first logical block represented by the LBA described in the first block device range descriptor is placed at the beginning of the data represented by the ROD token, and the first logical block represented by the LBA described in the second block device range descriptor is placed in the data represented by the ROD token immediately following the last logical block represented by the LBA described in the first block device range descriptor);
c) the logical blocks to be written from the data represented by the ROD token; and
d) the sequence of the logical blocks written from the data represented by the ROD token (e.g., the first logical block represented by the LBA described in the first block device range descriptor is written from the beginning of the data represented by the ROD token, and the first logical block represented by the LBA described in the second block device range descriptor is written from data represented by the ROD token immediately following the data written to the last logical block represented by the LBA described in the first block device range descriptor).

If the copy manager uses out of order transfers to create the representation of data for the ROD token, then the TRANSFER COUNT field in the parameter data in the response to a RECEIVE ROD TOKEN INFORMATION command with a list identifier that specifies a POPULATE TOKEN command (see 5.22.2) shall be based only on the contiguous transfers that complete without error starting at the first LBA specified by the first block device range descriptor (i.e., any transfers completed without error beyond the first incomplete or unsuccessful transfer shall not contribute to the computation of the value in the TRANSFER COUNT field).

If the copy manager uses out of order transfers to write from the data represented by the ROD token, then the TRANSFER COUNT field in the parameter data returned in response to a RECEIVE ROD TOKEN INFORMATION command with a list identifier that specifies a WRITE USING TOKEN command (see 5.22.3) shall be based only on the contiguous transfers that complete without error starting at the first LBA specified by the

first block device range descriptor (i.e., any transfers completed without error beyond the first incomplete or unsuccessful transfer shall not contribute to the computation of the value in the TRANSFER COUNT field).

### 4.30.2 POPULATE TOKEN command and WRITE USING TOKEN command completion

As part of completing a block device token operation originated by a POPULATE TOKEN command (see 5.7) or a WRITE USING TOKEN command (see 5.46), the copy manager shall compute the residual by subtracting the sum of the contents of the NUMBER OF LOGICAL BLOCK fields in all of the complete block device range descriptors of the parameter list (see 5.7.3) from the TRANSFER COUNT field in the parameter data returned in response to a RECEIVE ROD TOKEN INFORMATION command (see 5.22.2 and 5.22.3).

If the POPULATE TOKEN command was received with the immed bit set to zero, and the residual is negative, then the copy manager shall:

a) terminate the command with CHECK CONDITION status, with the additional sense code set to COPY TARGET DEVICE DATA UNDERRUN, the information field set to the transfer count, the valid bit set to one, and the sense key set to:
   A) COPY ABORTED, if the transfer count is not zero; or
   B) ILLEGAL REQUEST, if the transfer count is zero;
   or
b) complete the command with GOOD status and return parameter data for the RECEIVE ROD TOKEN INFORMATION command received on the same I_T nexus with a matching list identifier field with:
   A) the copy operation status field set to 03h or 60h (see SPC-4);
   B) the extended copy completion status field set to CHECK CONDITION (see SAM-5); and
   C) the sense data field with the additional sense code set to COPY TARGET DEVICE DATA UNDERRUN, the information field set to the transfer count, the valid bit set to one, and the sense key set to:
      a) COPY ABORTED, if the transfer count is not zero; or
      b) ILLEGAL REQUEST, if the transfer count is zero.

If the WRITE USING TOKEN command was received with the IMMED bit set to zero, and the residual is negative, then the copy manager shall terminate the command with CHECK CONDITION status, the additional sense code set to COPY TARGET DEVICE DATA UNDERRUN, the INFORMATION field set to the transfer count, the VALID bit set to one, and the sense key set to:

c) COPY ABORTED, if the transfer count is not zero; or
d) the sense key set to ILLEGAL REQUEST, if the transfer count is zero.

If the POPULATE TOKEN command or WRITE USING TOKEN command was received with the IMMED bit set to one, and the residual is negative, then the copy manager shall return parameter data for the RECEIVE ROD TOKEN INFORMATION command received on the same I_T nexus with a matching list identifier field with:

a) the copy operation status field set to 03h or 60h (see SPC-4);
b) the extended copy completion status field set to CHECK CONDITION status (see SAM-5); and
c) the sense data field with the additional sense code set to COPY TARGET DEVICE DATA UNDERRUN, the information field set to the transfer count, the valid bit set to one, and the sense key set to:
   A) COPY ABORTED, if the transfer count is not zero; or
   B) the sense key set to ILLEGAL REQUEST, if the transfer count is zero.

### 4.30.3 Block device specific ROD tokens

Block device specific ROD token types (see SPC-4) are shown in table 29.

**Table 29 — ROD token type values**

| ROD token type | Description | Reference |
|---|---|---|
| FF00_0000h to FFFF_0000h | Reserved | |
| FFFF_0001h | Block device zero ROD token | 4.30.4 |
| FFFF_0002h to FFFF_FFEFh | Reserved | |

### 4.30.4 Block device zero ROD token

The block device zero ROD token represents user data in which all bits are set to zero and protection information, if any, is set to FFFF_FFFF_FFFF_FFFFh. If user data with all bits set to zero and protection information, if any, set to FFFF_FFFF_FFFF_FFFFh, is represented by a ROD token, then, in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a POPULATE TOKEN command, the copy manager may or may not return a ROD token that is the block device zero ROD token. The block device zero ROD token format is shown in table 30.

**Table 30 — Block device zero ROD token format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | ROD TOKEN TYPE (FFFF_0001h) | | | | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | | | | | | | | |
| 5 | Reserved | | | | | | | |
| 6 | (MSB) | | | | | | | |
| 7 | ROD TOKEN LENGTH (01F8h) | | | | | | | (LSB) |
| 8 | | | | | | | | |
| ••• | Reserved | | | | | | | |
| 511 | | | | | | | | |

The block size associated with the block device zero ROD token is that of the block device to which the data is being written (e.g., if a block device zero ROD token is used to write to a logical unit that has 642-byte blocks, then the block size of the block device zero ROD token is 642 bytes).

The rod token type field is defined in SPC-4 and shall be set to the value shown in table 30 for the block device zero ROD token.

The rod token length field is defined in SPC-4 and shall be set to the value shown in table 30 for the block device zero ROD token.

### 4.30.5 ROD token device type specific data

The device type specific data for ROD tokens (see SPC-4) created by a copy manager for a direct access block devices (see 6.7):

a) provides information about the logical unit at the time that the ROD token was created; and
b) is a subset of the parameter data returned by the READ CAPACITY (16) command (see 5.16) for the logical unit that contains the copy manager that created the ROD token.

If the READ CAPACITY (16) parameter data changes so that the copy manager that created the ROD token is no longer able to access the data represented by the ROD token, then that copy manager shall invalidate the ROD token.

# 5 Commands for direct-access block devices

## 5.1 Commands for direct-access block devices overview

The commands for direct-access block devices are listed in table 31.

**Table 31 — Commands for direct-access block devices** (part 1 of 4)

| Command | Operation code [a] | Type | PI [b] | MACT [c] | Reference |
|---|---|---|---|---|---|
| ACCESS CONTROL IN | 86h | O | no | n/a | SPC-4 |
| ACCESS CONTROL OUT | 87h | O | no | n/a | SPC-4 |
| ATA PASS-THROUGH (12) | A1h | O | no | n/a | SAT-3 |
| ATA PASS-THROUGH (16) | 85h | O | no | n/a | SAT-3 |
| CHANGE ALIASES | A4h/0Bh | O | no | n/a | SPC-4 |
| COMPARE AND WRITE | 89h | O | yes | R, W | 5.2 |
| EXTENDED COPY | 83h | O | no | R, W | SPC-4 |
| FORMAT UNIT | 04h | M | yes | Z | 5.3 |
| GET LBA STATUS | 9Eh/12h | O | no | n/a | 5.4 |
| INQUIRY | 12h | M | yes | n/a | SPC-4 |
| LOG SELECT | 4Ch | O | no | n/a | SPC-4 |
| LOG SENSE | 4Dh | O | no | n/a | SPC-4 |
| MAINTENANCE IN | A3h/00h to 04h<br>A3h/06h to 09h | X | no | n/a | SPC-4 and SCC-2 |
| MAINTENANCE OUT | A4h/00h to 05h<br>A4h/07h to 09h | X | no | n/a | SPC-4 and SCC-2 |
| MODE SELECT (6) | 15h | O | no | n/a | SPC-4 |
| MODE SELECT (10) | 55h | O | no | n/a | SPC-4 |
| MODE SENSE (6) | 1Ah | O | no | n/a | SPC-4 |
| MODE SENSE (10) | 5Ah | O | no | n/a | SPC-4 |
| ORWRITE (16) | 8Bh | O | yes | R, W | 5.5 |
| ORWRITE (32) | 7F/000Eh | O | yes | R, W | 5.6 |
| PERSISTENT RESERVE IN | 5Eh | O | no | n/a | SPC-4 |
| PERSISTENT RESERVE OUT | 5Fh | O | no | n/a | SPC-4 |
| POPULATE TOKEN | 83h/10h | O | yes | n/a | 5.7 |
| PRE-FETCH (10) | 34h | O | no | R | 5.8 |
| PRE-FETCH (16) | 90h | O | no | R | 5.9 |
| PREVENT ALLOW MEDIUM REMOVAL | 1Eh | O | no | n/a | 5.10 |

[a] If a command is defined by a combination of operation code and service action, then the operation code value is shown preceding a slash and the service action value is shown after the slash.
[b] yes = commands with CDB or parameter list or parameter data fields that support protection information (see 4.22) or for which protection information may be a factor in the processing of the command.
[c] Logical block access command type (see 4.2.2): R = read, U = unmap, V = verify, W = write, Z = other.

**Table 31 — Commands for direct-access block devices** (part 2 of 4)

| Command | Operation code [a] | Type | PI [b] | MACT [c] | Reference |
|---|---|---|---|---|---|
| READ (10) | 28h | M | yes | R | 5.11 |
| READ (12) | A8h | O | yes | R | 5.12 |
| READ (16) | 88h | M | yes | R | 5.13 |
| READ (32) | 7Fh/0009h | O | yes | R | 5.14 |
| READ ATTRIBUTE | 8Ch | O | no | n/a | SPC-4 |
| READ BUFFER | 3Ch | O | no | n/a | SPC-4 |
| READ CAPACITY (10) | 25h | M | no | n/a | 5.15 |
| READ CAPACITY (16) | 9Eh/10h | M | yes | n/a | 5.16 |
| READ DEFECT DATA (10) | 37h | O | no | n/a | 5.17 |
| READ DEFECT DATA (12) | B7h | O | no | n/a | 5.18 |
| READ LONG (10) | 3Eh | O | yes | Z | 5.19 |
| READ LONG (16) | 9Eh/11h | O | yes | Z | 5.20 |
| REASSIGN BLOCKS | 07h | O | no | Z | 5.21 |
| RECEIVE COPY RESULTS | 84h | O | no | n/a | SPC-4 |
| RECEIVE DIAGNOSTIC RESULTS | 1Ch | X | no | n/a | SPC-4 |
| RECEIVE ROD TOKEN INFORMATION | 84h/07h | X | no | n/a | SPC-4 and 5.22 |
| REDUNDANCY GROUP IN | BAh | X | no | n/a | SCC-2 |
| REDUNDANCY GROUP OUT | BBh | X | no | n/a | SCC-2 |
| REPORT REFERRALS | 9Eh/13h | O | no | n/a | 5.23 |
| REPORT ALIASES | A3h/0Bh | O | no | n/a | SPC-4 |
| REPORT IDENTIFYING INFORMATION | A3h/05h | O | no | n/a | SPC-4 |
| REPORT LUNS | A0h | M | no | n/a | SPC-4 |
| REPORT PRIORITY | A3h/0Eh | O | no | n/a | SPC-4 |
| REPORT SUPPORTED OPERATION CODES | A3h/0Ch | O | no | n/a | SPC-4 |
| REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS | A3h/0Dh | O | no | n/a | SPC-4 |
| REPORT TARGET PORT GROUPS | A3h/0Ah | O | no | n/a | SPC-4 |
| REQUEST SENSE | 03h | M | no | n/a | SPC-4 |
| SANITIZE | 48h | O | no | Z | 5.24 |
| SECURITY PROTOCOL IN | A2h | O | no | n/a | SPC-4 |
| SECURITY PROTOCOL OUT | B5h | O | no | n/a | SPC-4 |

[a] If a command is defined by a combination of operation code and service action, then the operation code value is shown preceding a slash and the service action value is shown after the slash.
[b] yes = commands with CDB or parameter list or parameter data fields that support protection information (see 4.22) or for which protection information may be a factor in the processing of the command.
[c] Logical block access command type (see 4.2.2): R = read, U = unmap, V = verify, W = write, Z = other.

**Table 31 — Commands for direct-access block devices** (part 3 of 4)

| Command | Operation code [a] | Type | PI [b] | MACT [c] | Reference |
|---|---|---|---|---|---|
| SEND DIAGNOSTIC | 1Dh | M | no | n/a | SPC-4 |
| SET IDENTIFYING INFORMATION | A4h/06h | O | no | n/a | SPC-4 |
| SET PRIORITY | A4h/0Eh | O | no | n/a | SPC-4 |
| SET TARGET PORT GROUPS | A4h/0Ah | O | no | n/a | SPC-4 |
| SPARE IN | BCh | X | no | n/a | SCC-2 |
| SPARE OUT | BDh | X | no | n/a | SCC-2 |
| START STOP UNIT | 1Bh | O | no | n/a | 5.25 |
| SYNCHRONIZE CACHE (10) | 35h | O | no | W | 5.26 |
| SYNCHRONIZE CACHE (16) | 91h | O | no | W | 5.27 |
| TEST UNIT READY | 00h | M | no | n/a | SPC-4 |
| UNMAP | 42h | X | yes | U | 5.28 |
| VERIFY (10) | 2Fh | O | yes | V, W | 5.29 |
| VERIFY (12) | AFh | O | yes | V, W | 5.30 |
| VERIFY (16) | 8Fh | O | yes | V, W | 5.31 |
| VERIFY (32) | 7Fh/000Ah | O | yes | V, W | 5.32 |
| VOLUME SET IN | BEh | X | no | n/a | SCC-2 |
| VOLUME SET OUT | BFh | X | no | n/a | SCC-2 |
| WRITE (10) | 2Ah | O | yes | W | 5.33 |
| WRITE (12) | AAh | O | yes | W | 5.34 |
| WRITE (16) | 8Ah | O | yes | W | 5.35 |
| WRITE (32) | 7Fh/000Bh | O | yes | W | 5.36 |
| WRITE AND VERIFY (10) | 2Eh | O | yes | V, W | 5.37 |
| WRITE AND VERIFY (12) | AEh | O | yes | V, W | 5.38 |
| WRITE AND VERIFY (16) | 8Eh | O | yes | V, W | 5.39 |
| WRITE AND VERIFY (32) | 7Fh/000Ch | O | yes | V, W | 5.40 |
| WRITE ATTRIBUTE | 8Dh | O | no | n/a | SPC-4 |
| WRITE BUFFER | 3Bh | O | no | n/a | SPC-4 |
| WRITE LONG (10) | 3Fh | O | yes | Z | 5.41 |
| WRITE LONG (16) | 9Fh/11h | O | yes | Z | 5.42 |
| WRITE SAME (10) | 41h | O | yes | U, W | 5.43 |
| WRITE SAME (16) | 93h | X | yes | U, W | 5.44 |

[a] If a command is defined by a combination of operation code and service action, then the operation code value is shown preceding a slash and the service action value is shown after the slash.

[b] yes = commands with CDB or parameter list or parameter data fields that support protection information (see 4.22) or for which protection information may be a factor in the processing of the command.

[c] Logical block access command type (see 4.2.2): R = read, U = unmap, V = verify, W = write, Z = other.

**Table 31 — Commands for direct-access block devices** (part 4 of 4)

| Command | Operation code [a] | Type | PI [b] | MACT [c] | Reference |
|---|---|---|---|---|---|
| WRITE SAME (32) | 7Fh/000Dh | X | yes | U, W | 5.45 |
| WRITE USING TOKEN | 83h/11h | X | yes | Z | 5.46 |
| XDWRITEREAD (10) | 53h | O | yes | R, W | 5.47 |
| XDWRITEREAD (32) | 7Fh/0007h | O | yes | R, W | 5.48 |
| XPWRITE (10) | 51h | O | yes | R, W | 5.49 |
| XPWRITE (32) | 7Fh/0006h | O | yes | R, W | 5.50 |

Notes:

a) The following operation codes are obsolete: 01h, 08h, 0Ah, 0Bh, 16h, 17h, 18h, 2Bh, 30h, 31h, 32h, 33h, 36h, 39h, 3Ah, 40h, 50h, 52h, 56h, 57h, 80h, 81h, 82h, 92h, A7h, B3h, and B4h.

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

c) A complete summary of operation codes is available at http://www.t10.org/lists/2op.htm. The summary includes information about obsolete commands.

[a] If a command is defined by a combination of operation code and service action, then the operation code value is shown preceding a slash and the service action value is shown after the slash.

[b] yes = commands with CDB or parameter list or parameter data fields that support protection information (see 4.22) or for which protection information may be a factor in the processing of the command.

[c] Logical block access command type (see 4.2.2): R = read, U = unmap, V = verify, W = write, Z = other.

### 5.2 COMPARE AND WRITE command

The COMPARE AND WRITE command (see table 32) requests that the device server perform the following as an uninterrupted series of actions (see 4.27):

1) if required by the rules for caching (see 4.15), then perform read medium operations on the logical blocks referenced by the LBAs specified by this command;
2) perform a compare operation on only the user data (i.e., not on the protection information) from:
   A) the referenced logical blocks; and
   B) the compare instance transferred from the Data-Out Buffer;
3) if the compare operation indicates a match, then perform the following operations:
   1) check the protection information, if any, in the write instance of the logical blocks transferred from the Data-Out Buffer based on the contents of the WRPROTECT field as defined in table 109; and
   2) write the write instance of the logical blocks to the LBAs specified by this command based on the rules for caching;

   and
4) if the compare operation does not indicate a match, then the device server terminates the command with:
   A) CHECK CONDITION status;
   B) the sense key set to MISCOMPARE;
   C) the additional sense code set to MISCOMPARE DURING VERIFY OPERATION;
   D) the INFORMATION field in the sense data set to the offset from the start of the Data-Out Buffer to the first byte of data that was not equal; and
   E) the VALID bit in the sense data set to one.

The Data-Out Buffer contains two instances of logical block data:

1) the compare instance, in which:
   A) the user data is used for the compare operation; and
   B) the protection information, if any, is not used;

   and
2) the write instance, in which:
   A) the user data is used for the write medium operation; and
   B) the protection information, if any, is used for the write medium operation.

**Table 32 — COMPARE AND WRITE command**

| Bit / Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (89h) | | | | | | | |
| 1 | WRPROTECT | | | DPO | FUA | Reserved | FUA_NV | Reserved |
| 2 | (MSB) | | | | | | | |
| ••• | LOGICAL BLOCK ADDRESS | | | | | | | |
| 9 | | | | | | | | (LSB) |
| 10 | | | | | | | | |
| ••• | Reserved | | | | | | | |
| 12 | | | | | | | | |
| 13 | NUMBER OF LOGICAL BLOCKS | | | | | | | |
| 14 | Reserved | | | GROUP NUMBER | | | | |
| 15 | CONTROL | | | | | | | |

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 32 for the COMPARE AND WRITE command.

See the READ (10) command (see 5.11) for the definition of the DPO bit. See the READ (10) command (see 5.11) for the definitions of the FUA bit and the FUA_NV bit specifying behavior for read medium operations. See the WRITE (10) command (see 5.33) for the definitions of the FUA bit and the FUA_NV bit specifying behavior for write medium operations.

See the WRITE (10) command for the definition of the WRPROTECT field. If the Verify Error Recovery mode page (see 6.5.9) is implemented, then the current settings in that page specify the criteria used when a read medium operation is specified. If the Verify Error Recovery mode page is not implemented, then the verification criteria are vendor specific.

The LOGICAL BLOCK ADDRESS field specifies the LBA of the first logical block (see 4.5) to be accessed by the device server (e.g., the LBA of the first logical block accessed by the read medium operations and the LBA of the first logical block accessed by the write medium operations). If the specified LBA exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

The NUMBER OF LOGICAL BLOCKS field specifies:

a) the number of contiguous logical blocks on which read medium operations shall be performed based on the rules for caching, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field;
b) the number of contiguous logical blocks that shall be transferred from the Data-Out Buffer for the compare operation; and
c) if the compare operation indicates a match, then the number of contiguous logical blocks that shall be transferred from the Data-Out Buffer and on which write medium operations shall be performed based on the rules for caching, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field.

A NUMBER OF LOGICAL BLOCKS field set to zero specifies that no logical blocks shall be read from medium, no data shall be transferred and compared, and no logical blocks shall be written to the medium. This condition shall not be considered an error.

If the specified LBA and the specified number of logical blocks exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

If the number of logical blocks exceeds the value in the MAXIMUM COMPARE AND WRITE LENGTH field in the Block Limits VPD page (see 6.6.3), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

See the PRE-FETCH (10) command (see 5.8) for the definition of the GROUP NUMBER field.

The CONTROL byte is defined in SAM-5.

## 5.3 FORMAT UNIT command

### 5.3.1 FORMAT UNIT command overview

The FORMAT UNIT command (see table 33) requests that the device server format the medium into application client accessible logical blocks as specified in the number of logical blocks and logical block length values received in the last mode parameter block descriptor (see 6.5.2) in a MODE SELECT command (see SPC-4). In addition, the device server may certify the medium and create control structures for the management of the medium and defects. The degree that the medium is altered by this command is vendor specific.

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

If any deferred downloaded code has been received as a result of a WRITE BUFFER command with the MODE field set to 0Eh (see SPC-4), then that deferred downloaded code shall replace the current operational code.

Before performing the operation specified by this command, the device server shall stop all:

a) enabled power condition timers (see SPC-4):
b) timers for enabled background scan operations (see 4.24); and
c) timers or counters enabled for device-specific background functions.

After the operation is complete, the device server shall reinitialize and restart all enabled timers and counters for power conditions and background functions.

While performing a format operation, the device server shall terminate all commands, except INQUIRY commands, REPORT LUNS commands, and REQUEST SENSE commands, with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, FORMAT IN PROGRESS. If the device server receives an INQUIRY command, a REPORT LUNS command, or a REQUEST SENSE command, then the device server shall process the command. The device server shall return data for an INQUIRY command based on the condition of the logical unit before beginning the FORMAT UNIT command (i.e., INQUIRY data shall not change until after successful completion of a format operation). The processing of commands already in a task set when a FORMAT UNIT command is received is vendor specific.

The PROGRESS INDICATION field in parameter data returned by the device server in response to a REQUEST SENSE command (see SPC-4) may be used by the application client at any time during a format operation to poll the logical unit's progress. While a format operation is in progress unless an error has occurred, a device server shall respond to a REQUEST SENSE command by returning parameter data containing sense data with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, FORMAT IN PROGRESS with the sense key specific bytes set for progress indication (see SPC-4).

The simplest form of the FORMAT UNIT command (i.e., a FORMAT UNIT command with no parameter data) accomplishes medium formatting with little application client control over defect management. The device server implementation determines the degree of defect management that is to be performed. Additional forms of this command increase the application client's control over defect management (see 4.13). The application client may specify:

a) that the device server clear the existing GLIST;
b) a list of address descriptors that the device server adds to the GLIST;
c) that the device server enable a certification operation that adds address descriptors for physical blocks with medium defects discovered during the certification operation to the GLIST; and
d) the behavior of the device server if it is not able to access the PLIST or GLIST or determine whether one or both of them exists.

Following a successful format operation:

a) if the logical unit is fully provisioned (i.e., the LBPME bit in the READ CAPACITY (16) parameter data is set to zero), then all LBAs in the logical unit are mapped (see 4.7.2); or
b) if the logical unit supports logical block provisioning management (i.e., the LBPME bit is set to one), then:
    A) if the LBPRZ bit in the READ CAPACITY (16) parameter data is set to zero, then each LBA in the logical unit shall be either:
        a) mapped, if an initialization pattern was specified that does not match the vendor-specific data returned by a read command for an unmapped LBA (see 4.7.4.2); or
        b) unmapped (see 4.7.3.2 and 4.7.3.3), if no initialization pattern was specified or an initialization pattern was specified that matches the vendor-specific data returned by a read command for an unmapped LBA (see 4.7.4.2); and
    B) if the LBPRZ bit is set to one, then each LBA in the logical unit:
        a) shall be mapped, if the format operation did not initialize the user data to all zeroes for the logical block referenced by that LBA;
        b) shall be unmapped (see 4.7.3.2 and 4.7.3.3), if the format operation initialized the user data to all zeroes for the logical blocks referenced by all valid LBAs in the logical unit; or

c) may be unmapped (see 4.7.3.2 and 4.7.3.3), if the format operation initialized the user data to all zeroes for the logical block referenced by that LBA, and the format operation did not initialize the user data to all zeroes for the logical blocks referenced by all valid LBAs in the logical unit.

Table 33 defines the FORMAT UNIT command.

**Table 33 — FORMAT UNIT command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 33 for the FORMAT UNIT command.

The combination (see table 38) of the format protection information (FMTPINFO) field and the PROTECTION FIELD USAGE field in the parameter list header (see 5.3.2.2) specifies whether or not the device server enables or disables the use of protection information.

A LONGLIST bit set to zero specifies that the parameter list, if any, contains a short parameter list header as defined in table 36. A LONGLIST bit set to one specifies that the parameter list, if any, contains a long parameter list header as defined in table 37. If the FMTDATA bit is set to zero, then the LONGLIST bit shall be ignored.

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

A FMTDATA bit set to one specifies that the FORMAT UNIT parameter list (see 5.3.2) shall be transferred from the Data-Out Buffer.

If the FMTDATA bit is set to zero and the FMTPINFO field is not set to zero, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

Following a successful format operation, the PROT_EN bit and the P_TYPE field in the READ CAPACITY (16) parameter data (see 5.16.2) indicate the type of protection currently in effect on the logical unit.

If protection information is written during a format operation (i.e., the FMTPINFO field is set to a value greater than zero), then protection information shall be written to a default value of FFFF_FFFF_FFFF_FFFFh.

A complete list (CMPLST) bit set to zero specifies that the device server shall add the defect list included in the FORMAT UNIT parameter list to the existing GLIST (see 4.13). A CMPLST bit set to one specifies that the device server shall replace the existing GLIST with the defect list, if any, included in the FORMAT UNIT parameter list.

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

If the FMTDATA bit is set to one, then the DEFECT LIST FORMAT field specifies the format of the address descriptors in the defect list in the FORMAT UNIT parameter list.

Table 34 defines support requirements for address descriptors based on the combinations of the FMTDATA bit, the CMPLST bit, the DEFECT LIST FORMAT field, and the DEFECT LIST LENGTH field.

**Table 34 — FORMAT UNIT command address descriptor support requirements**

| Field in the FORMAT UNIT CDB | | | DEFECT LIST LENGTH [a] | Support | Comments |
|---|---|---|---|---|---|
| FMTDATA | CMPLST | DEFECT LIST FORMAT | | | |
| 0 | any | 000b | Not available | M | Vendor specific defect information |
| 1 | 0 | Either:<br>a) 000b (i.e., short block address descriptor)(see 6.2.2);<br>b) 001b (i.e., extended bytes from index address descriptor)(see 6.2.3);<br>c) 010b (i.e., extended physical sector address descriptor)(see 6.2.4);<br>d) 011b (i.e., long block address descriptor)(see 6.2.5);<br>e) 100b (i.e., bytes from index address descriptor)(see 6.2.6); or<br>f) 101b (i.e., physical sector address descriptor)(see 6.2.7) | Zero | O | See [b] and [d] |
| | 1 | | | O | See [b] and [e] |
| | 0 | | Nonzero | O | See [c] and [d] |
| | 1 | | | O | See [c] and [e] |
| | 0 | 110b (i.e., vendor specific) | Zero | O | See [b] |
| | | | Nonzero | O | See [c] |
| | 1 | | Zero | O | See [b] |
| | | | Nonzero | O | See [c] |
| All other combinations | | | | Reserved | |

[a] This field is in the parameter list header.
[b] No defect list is included in the parameter list.
[c] A defect list is included in the parameter list.
[d] The device server retains the existing GLIST.
[e] The device server discards the existing GLIST.

The CONTROL byte is defined in SAM-5.

### 5.3.2 FORMAT UNIT parameter list

#### 5.3.2.1 FORMAT UNIT parameter list overview

Table 35 defines the FORMAT UNIT parameter list.

**Table 35 — FORMAT UNIT parameter list**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 to 3 or 0 to 7 | Parameter list header (see table 36 or table 37 in 5.3.2.2) | | | | | | | |
| | Initialization pattern descriptor (if any) (see table 39 in 5.3.2.3) | | | | | | | |
| | Defect list (if any) | | | | | | | |

The parameter list header is defined in 5.3.2.2.

The initialization pattern descriptor, if any, is defined in 5.3.2.3.

The defect list, if any, contains address descriptors (see 6.2) each specifying a location on the medium to which the device server shall not assign LBAs. The device server shall maintain the current logical block to physical block alignment (see 4.6) for logical blocks not specified in the defect list.

Short block format address descriptors and long block format address descriptors should be in ascending order. Bytes from index format address descriptors and physical sector format address descriptors shall be in ascending order. If the address descriptors are not in the required order, then the device server may terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

#### 5.3.2.2 Parameter list header

The parameter list headers (see table 36 and table 37) provide several optional format control parameters. If the application client requests a function that is not implemented by the device server, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the LONGLIST bit is set to zero in the FORMAT UNIT CDB, then the short parameter list header (see table 36) is used.

**Table 36 — Short parameter list header**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | | | | | PROTECTION FIELD USAGE | | |
| | FOV | Format options bits | | | | Obsolete | IMMED | Vendor specific |
| 1 | | DPRY | DCRT | STPF | IP | | | |
| 2 | (MSB) DEFECT LIST LENGTH | | | | | | | |
| 3 | | | | | | | | (LSB) |

If the LONGLIST bit is set to one in the FORMAT UNIT CDB, then the long parameter list header (see table 37) is used.

**Table 37 — Long parameter list header**

| Bit / Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | | | | | PROTECTION FIELD USAGE | | |
| | FOV | Format options bits | | | | Obsolete | IMMED | Vendor specific |
| 1 | | DPRY | DCRT | STPF | IP | | | |
| 2 | Reserved | | | | | | | |
| 3 | P_I_INFORMATION | | | | PROTECTION INTERVAL EXPONENT | | | |
| 4 | (MSB) | DEFECT LIST LENGTH | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | (LSB) |

The combination (see table 38) of the PROTECTION FIELD USAGE field and the FMTPINFO field (see 5.3.1) specifies the requested protection type (see 4.22.2).

**Table 38 — FMTPINFO field and PROTECTION FIELD USAGE field** (part 1 of 2)

| Fields indicated by the device server | | Fields specified by the application client | | Description |
|---|---|---|---|---|
| SPT [a] | PROTECT [b] | FMTPINFO | PROTECTION FIELD USAGE | |
| xxxb | 0b | 00b | 000b | The logical unit shall be formatted to type 0 protection [c] (see 4.22.2.2) resulting in the PROT_EN field [d] being set to 000b. |
| | | | >000b | Illegal [e] |
| | | 01b | xxxb | Illegal [f] |
| | | 1xb | xxxb | Illegal [f] |

[a] See the Extended INQUIRY Data VPD page (see SPC-4) for the definition of the SPT field.
[b] See the standard INQUIRY data (see SPC-4) for the definition of the PROTECT bit.
[c] The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header (see SPC-4).
[d] See the READ CAPACITY (16) parameter data (see 5.16.2) for the definition of the P_TYPE field.
[e] The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.
[f] The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
[g] The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header plus eight bytes for each protection information interval (e.g., if the logical block length is 2 048 and the PROTECTION INTERVAL EXPONENT field set to two, then there are four 512 byte protection information intervals each followed by eight bytes of protection information resulting in a formatted logical block length of 2 080 bytes). Following a successful format operation, the PROT_EN bit in the READ CAPACITY (16) parameter data (see 5.16.2) indicates whether protection information (see 4.22) is enabled.

**Table 38 — FMTPINFO field and PROTECTION FIELD USAGE field** (part 2 of 2)

| Fields indicated by the device server | | Fields specified by the application client | | Description |
|---|---|---|---|---|
| SPT [a] | PROTECT [b] | FMTPINFO | PROTECTION FIELD USAGE | |
| xxxb | 1b | 00b | 000b | The logical unit shall be formatted to type 0 protection [c] (see 4.22.2.2) resulting in the PROT_EN field [d] being set to 000b. |
| | | | >000b | Illegal [e] |
| xxxb | 1b | 01b | xxxb | Illegal [f] |
| 000b<br>001b<br>011b<br>111b | 1b | 10b | 000b | The logical unit shall be formatted to type 1 protection [g] (see 4.22.2.3) resulting in the P_TYPE field [d] being set to 000b. |
| | | | >000b | Illegal [e] |
| 010b<br>100b<br>101b | 1b | 10b | xxxb | Illegal [f] |
| 000b | 1b | 11b | xxxb | Illegal [f] |
| 001b<br>010b<br>101b<br>111b | 1b | 11b | 000b | The logical unit shall be formatted to type 2 protection [g] (see 4.22.2.4) resulting in the P_TYPE field [d] being set to 001b. |
| 001b<br>010b | 1b | 11b | >000b | Illegal [e] |
| 011b<br>100b | 1b | 11b | 000b | Illegal [e] |
| 011b<br>100b<br>101b<br>111b | 1b | 11b | 001b | The logical unit shall be formatted to type 3 protection. [g] (see 4.22.2.5) resulting in the P_TYPE field [d] being set to 010b. |
| | | | >001b | Illegal [e] |
| 110b | 1b | 10b<br>11b | xxxb | Reserved |

[a] See the Extended INQUIRY Data VPD page (see SPC-4) for the definition of the SPT field.
[b] See the standard INQUIRY data (see SPC-4) for the definition of the PROTECT bit.
[c] The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header (see SPC-4).
[d] See the READ CAPACITY (16) parameter data (see 5.16.2) for the definition of the P_TYPE field.
[e] The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.
[f] The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
[g] The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header plus eight bytes for each protection information interval (e.g., if the logical block length is 2 048 and the PROTECTION INTERVAL EXPONENT field set to two, then there are four 512 byte protection information intervals each followed by eight bytes of protection information resulting in a formatted logical block length of 2 080 bytes). Following a successful format operation, the PROT_EN bit in the READ CAPACITY (16) parameter data (see 5.16.2) indicates whether protection information (see 4.22) is enabled.

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

A FOV bit set to one specifies that the device server shall process the format options bits as follows:

a) a disable primary (DPRY) bit:
   A) set to zero specifies that the device server shall not assign LBAs to parts of the medium identified as defective in the PLIST; or
   B) set to one specifies that the device server shall not use the PLIST to identify defective areas of the medium, and the PLIST shall not be deleted;
b) a disable certification (DCRT) bit:
   A) set to zero specifies that the device server shall perform a vendor specific medium certification operation and add address descriptors for defects that it detects during the certification operation to the GLIST; or
   B) set to one specifies that the device server shall not perform any vendor specific medium certification process or format verification operation;
c) the stop format (STPF) bit controls the behavior of the device server if the device server has been requested to use the PLIST (i.e., the DPRY bit is set to zero) or the GLIST (i.e., the CMPLST bit is set to zero) and one or more of the following occurs:
   A) list locate error: the device server is not able to locate a specified defect list or determine whether a specified defect list exists; or
   B) list access error: the device server encounters an error while accessing a specified defect list;
d) a STPF bit set to zero specifies that:
   A) if a list locate error or a list access error occurs, then the device server shall continue to process the FORMAT UNIT command; and
   B) after the format operation is complete, if:
      a) a list locate error and a list access error both occurred, then the device server shall terminate the FORMAT UNIT command with CHECK CONDITION status at the completion of the command with the sense key set to RECOVERED ERROR and the additional sense code set to DEFECT LIST NOT FOUND or DEFECT LIST ERROR;
      b) a list locate error occurred and a list access error did not occur, then the device server shall terminate the FORMAT UNIT command with CHECK CONDITION status with the sense key set to RECOVERED ERROR with the additional sense code set to DEFECT LIST NOT FOUND; and
      c) a list access error occurred and a list locate error did not occur, then the device server shall terminate the FORMAT UNIT command with CHECK CONDITION status with the sense key set to RECOVERED ERROR with the additional sense code set to DEFECT LIST ERROR;
e) a STPF bit set to one specifies that:
   A) if a list locate error occurs, then the device server shall terminate the FORMAT UNIT command with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to either DEFECT LIST NOT FOUND; or
   B) if a list access error occurs, then the device server shall terminate the FORMAT UNIT command with CHECK CONDITION status with the sense key set to MEDIUM ERROR with the additional sense code set to DEFECT LIST ERROR;

   and
f) an initialization pattern (IP) bit:
   A) set to zero specifies that an initialization pattern descriptor (see 5.3.2.3) is not included and that the device server shall use its default initialization pattern; or
   B) set to one specifies that:
      a) an initialization pattern descriptor is included in the FORMAT UNIT parameter list following the parameter list header; and
      b) if the device server does not support initialization pattern descriptors, then the device server shall terminate the FORMAT UNIT command with CHECK CONDITION status with the sense

key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

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

The P_I_INFORMATION field, if any (i.e., if the long parameter list header is used), shall be set to 0h.

For a type 1 protection information request, if the PROTECTION INTERVAL EXPONENT field, if any, is not set to 0h, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

For a type 2 protection or a type 3 protection format request, the protection interval exponent determines the length of user data to be sent before protection information is transferred (i.e., the protection information interval).

The protection information interval is calculated as follows:

protection information interval = logical block length ÷ $2^{(\text{protection interval exponent})}$

where:

| | |
|---|---|
| logical block length | is the number of bytes of user data in a logical block (see 4.5) |
| protection interval exponent | is zero if the short parameter list header (see table 36) is used or the contents of the PROTECTION INTERVAL EXPONENT field if the long parameter list header (see table 37) is used |

If the protection information interval calculates to a value that is not an even number (e.g., 520 ÷ $2^3$ = 65) or not a whole number (e.g., 520 ÷ $2^4$ = 32.5 and 520 ÷ $2^{10}$ = 0.508), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The DEFECT LIST LENGTH field specifies the total length in bytes of the defect list (i.e., the address descriptors) that follows and does not include the length of the initialization pattern descriptor, if any. The formats for the address descriptor(s) are shown in 6.2.

#### 5.3.2.3 Initialization pattern descriptor

The initialization pattern descriptor specifies that the device server initialize logical blocks to a specified pattern. The initialization pattern descriptor (see table 39) is sent to the device server as part of the FORMAT UNIT parameter list.

**Table 39 — Initialization pattern descriptor**

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

A security initialize (SI) bit set to one specifies that the device server shall attempt to write the initialization pattern to all areas of the medium including those that may have been reassigned (i.e., are in a defect list). An SI bit set to one shall take precedence over any other FORMAT UNIT CDB field. The initialization pattern shall be written using a security erasure write technique. Application clients may use this command multiple times

to fully erase the previous data. Such security erasure write technique procedures are outside the scope of this standard. The exact requirements placed on the security erasure write technique are vendor specific. The intent of the security erasure write is to render any previous user data unrecoverable by any analog or digital technique.

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

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

NOTE 7 - Migration from the SI bit to the SANITIZE command (see 5.24) is recommended for all implementations.

The INITIALIZATION PATTERN TYPE field (see table 40) specifies the type of pattern the device server shall use to initialize each logical block within the application client accessible part of the medium. All bytes within a logical block shall be written with the initialization pattern.

**Table 40 — INITIALIZATION PATTERN TYPE field**

| Code | Description |
|---|---|
| 00h | Use a default initialization pattern [a] |
| 01h | Repeat the pattern specified in the INITIALIZATION PATTERN field as required to fill the logical block [b] |
| 02h to 7Fh | Reserved |
| 80h to FFh | Vendor specific |

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

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

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

The INITIALIZATION PATTERN field specifies the initialization pattern.

## 5.4 GET LBA STATUS command

### 5.4.1 GET LBA STATUS command overview

The GET LBA STATUS command (see table 41) requests that the device server transfer parameter data describing the logical block provisioning status (see 4.7) for the specified LBA and a vendor specific number of subsequent LBAs to the Data-In Buffer.

The device server may or may not process this command as an uninterrupted sequence of actions (e.g., if concurrent operations are occurring that affect the logical block provisioning status, then the returned parameter data may be inconsistent or out of date).

**Table 41 — GET LBA STATUS command**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (9Eh) | | | | | | | |
| 1 | Reserved | | | SERVICE ACTION (12h) | | | | |
| 2 | (MSB) | | | | | | | |
| ••• | STARTING LOGICAL BLOCK ADDRESS | | | | | | | |
| 9 | | | | | | | | (LSB) |
| 10 | (MSB) | | | | | | | |
| ••• | ALLOCATION LENGTH | | | | | | | |
| 13 | | | | | | | | (LSB) |
| 14 | Reserved | | | | | | | |
| 15 | CONTROL | | | | | | | |

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 41 for the GET LBA STATUS command.

The SERVICE ACTION field is defined in SPC-4 and shall be set to the value shown in table 41 for the GET LBA STATUS command.

The STARTING LOGICAL BLOCK ADDRESS field specifies the LBA of the first logical block addressed by this command. If the specified starting LBA exceeds the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

The ALLOCATION LENGTH field is defined in SPC-4. In response to a GET LBA STATUS command, the device server may send less data to the Data-In Buffer than is specified by the allocation length. If, in response to a single GET LBA STATUS command, the device server does not send sufficient data to the Data-In Buffer to satisfy the requirement of the application client, then, to retrieve additional information, the application client may send additional GET LBA STATUS commands with different starting LBA values.

The CONTROL byte is defined in SAM-5.

### 5.4.2 GET LBA STATUS parameter data

#### 5.4.2.1 GET LBA STATUS parameter data overview

The GET LBA STATUS parameter data (see table 42) contains an eight-byte header followed by one or more LBA status descriptors.

**Table 42 — GET LBA STATUS parameter data**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | | | | PARAMETER DATA LENGTH (n - 3) | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | | | | | | | | |
| ••• | | | | Reserved | | | | |
| 7 | | | | | | | | |
| **LBA status descriptors** | | | | | | | | |
| 8 | | | | | | | | |
| ••• | | | | LBA status descriptor [first] (see 5.4.2.2) | | | | |
| 23 | | | | | | | | |
| | | | | ⋮ | | | | |
| n - 15 | | | | | | | | |
| ••• | | | | LBA status descriptor [last] (see 5.4.2.2) | | | | |
| n | | | | | | | | |

The PARAMETER DATA LENGTH field indicates the number of bytes of parameter data that follow. The value in the PARAMETER DATA LENGTH field shall be:

a) at least 20 (i.e., the available parameter data shall contain at least one LBA status descriptor); and
b) four added to a multiple of 16 (i.e., the available parameter data shall end on a boundary between LBA Status descriptors).

Due to processing considerations outside the scope of this standard, two GET LBA STATUS commands with identical values in all CDB fields may result in two different values in the PARAMETER DATA LENGTH field.

The relationship between the PARAMETER DATA LENGTH field and the ALLOCATION LENGTH field in the CDB is defined in SPC-4.

### 5.4.2.2 LBA status descriptor

The LBA status descriptor (see table 43) contains LBA status information for one or more LBAs.

**Table 43 — LBA status descriptor format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | LBA STATUS LOGICAL BLOCK ADDRESS | | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | (MSB) | | | | | | | |
| ••• | NUMBER OF LOGICAL BLOCKS | | | | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | Reserved | | | | PROVISIONING STATUS | | | |
| 13 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |

The LBA STATUS LOGICAL BLOCK ADDRESS field contains the first LBA of the LBA extent for which this descriptor reports LBA status.

The NUMBER OF LOGICAL BLOCKS field contains the number of logical blocks in that extent. The device server should return the largest possible value in the NUMBER OF LOGICAL BLOCKS field.

The PROVISIONING STATUS field is defined in table 44.

**Table 44 — PROVISIONING STATUS field**

| Code | Description |
|---|---|
| 0h | Each LBA in the LBA extent is mapped (see 4.7.4.3) has an unknown status. |
| 1h | Each LBA in the LBA extent is deallocated (see 4.7.4.4). |
| 2h | Each LBA in the LBA extent is anchored (see 4.7.4.5). |
| All others | Reserved |

If the logical unit is fully provisioned (see 4.7.2), then the PROVISIONING STATUS field for all LBAs shall be set to 0h (i.e., mapped or unknown).

### 5.4.2.3 LBA status descriptor relationships

The LBA STATUS LOGICAL BLOCK ADDRESS field in the first LBA status descriptor returned in the GET LBA STATUS parameter data shall contain the value specified in the STARTING LOGICAL BLOCK ADDRESS field of the CDB. For subsequent LBA status descriptors, the contents of the LBA STATUS LOGICAL BLOCK ADDRESS field shall contain the sum of the values in:

a) the LBA STATUS LOGICAL BLOCK ADDRESS field in the previous LBA status descriptor; and
b) the NUMBER OF LOGICAL BLOCKS field in the previous LBA status descriptor.

Adjacent LBA status descriptors may have the same values for the PROVISIONING STATUS field.

### 5.5 ORWRITE (16) command

The ORWRITE (16) command (see table 45) requests that the device server perform an ORWRITE command (see 4.29) set operation (see 4.29.4).

**Table 45 — ORWRITE (16) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 45 for the ORWRITE (16) command.

See the READ (10) command (see 5.11) for the definitions of the FUA bit and the FUA_NV bit specifying behavior for read medium operations. See the WRITE (10) command (see 5.33) for the definitions of the FUA bit and the FUA_NV bit specifying behavior for write medium operations. See the READ (10) command (see 5.11) for the definition of the DPO bit. See the PRE-FETCH (10) command (see 5.8) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.8) and 4.23 for the definition of the GROUP NUMBER field.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that are read, transferred from the Data-Out Buffer, and ORed into a bitmap buffer (see 3.2.9), starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. If the specified LBA and the specified transfer length exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.6.3).

The CONTROL byte is defined in SAM-5.

The device server shall:

a) check protection information read from the medium based on the ORPROTECT field as described in table 46; and
b) check protection information transferred from the Data-Out Buffer based on the ORPROTECT field as described in table 47.

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

The device server shall check the protection information read from the medium based on the ORPROTECT field as described in table 46. All footnotes for table 46 are at the end of the table.

**Table 46 — ORPROTECT field - checking protection information read from the medium** (part 1 of 3)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d e], additional sense code |
|---|---|---|---|---|
| 000b | Yes [i j] | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | GRD_CHK = 0 | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | No protection information on the medium to check. Only user data is checked. | | |
| 001b<br>101b [b] | Yes | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | GRD_CHK = 0 | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | Error condition [a] | | |
| 010b [b] | Yes | LOGICAL BLOCK GUARD | No check performed | |
| | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | Error condition [a] | | |

**Table 46 — ORPROTECT field - checking protection information read from the medium** (part 2 of 3)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d e], additional sense code |
|---|---|---|---|---|
| 011b [b] | Yes | LOGICAL BLOCK GUARD | No check performed | |
| | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No | Error condition [a] | | |
| 100b [b] | Yes | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | GRD_CHK = 0 | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No | Error condition [a] | | |

**Table 46 — ORPROTECT field - checking protection information read from the medium** (part 3 of 3)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d e], additional sense code |
|---|---|---|---|---|
| 110b to 111b | Reserved | | | |

[a] If an ORWRITE command is sent to a logical unit that supports protection information (see 4.22), and the logical unit has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information, then the device server should terminate the ORWRITE command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[c] If the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field, then the device server shall check the logical block application tag. If the ATMPE bit in the Control mode page (see SPC-4) is set to one, then this knowledge is acquired from the Application Tag mode page. If the ATMPE bit is set to zero, then the method for acquiring this knowledge is not defined by this standard.

[d] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

[e] If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

[f] See the Extended INQUIRY Data VPD page (see SPC-4) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bit.

[g] If the device server detects:

a) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, and type 1 protection (see 4.22.2.3) is enabled; or

b) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, the LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.22.2.5) is enabled,

then the device server shall not check any protection information in the associated protection information interval.

[h] If type 1 protection is enabled, then the device server shall check the logical block reference tag by comparing it to the lower four bytes of the LBA associated with the logical block. If type 3 protection is enabled, then the device server shall check each logical block reference tag only if the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. The method for acquiring this knowledge is not defined by this standard.

[i] If the RWWP bit in the Control mode page (see SPC-4) is set to one, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[j] If the DPICZ bit in the Control mode page (see SPC-4) is set to one, and the RWWP bit in the Control mode page is set to zero, then protection information shall not be checked.

The device server shall check the protection information transferred from the Data-Out Buffer based on the ORPROTECT field as described in table 47. All footnotes for table 47 are at the end of the table.

**Table 47 — ORPROTECT field - checking protection information from the Data-Out Buffer** (part 1 of 2)

| Code | Logical unit formatted with protection information | Field in protection information | Device server check | If check fails [d] [e], additional sense code |
|---|---|---|---|---|
| 000b | Yes | No protection information received from application client to check | | |
| | No | No protection information received from application client to check | | |
| 001b [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | Dependent on RWWP [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | Shall (except for type 3) [f] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [a] | | |
| 010b [b] | Yes | LOGICAL BLOCK GUARD | Shall not | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | Dependent on RWWP [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | May [f] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [a] | | |
| 011b [b] | Yes | LOGICAL BLOCK GUARD | Shall not | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | Shall not | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | Shall not | No check performed |
| | No | Error condition [a] | | |
| 100b [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | Shall not | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | Shall not | No check performed |
| | No | Error condition [a] | | |
| 101b [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | Dependent on RWWP [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | May [f] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [a] | | |

**Table 47 — ORPROTECT field - checking protection information from the Data-Out Buffer** (part 2 of 2)

| Code | Logical unit formatted with protection information | Field in protection information | Device server check | If check fails [d] [e], additional sense code |
|---|---|---|---|---|
| 110b to 111b | Reserved | | | |

[a] If an ORWRITE command is sent to a logical unit that supports protection information (see 4.22), and the logical unit has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information, then the device server should terminate the ORWRITE command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[c] If the ATO bit is set to one in the Control mode page (see SPC-4), and the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field, then the device server:

a) may check each logical block application tag if the RWWP bit is set to zero in the Control mode page (see SPC-4); and

b) shall check each logical block application tag if the RWWP bit is set to one in the Control mode page.

If the ATMPE bit in the Control mode page (see SPC-4) is set to one, then this knowledge is acquired from the Application Tag mode page. If the ATMPE bit is set to zero, then the method for acquiring this knowledge is not defined by this standard.

[d] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

[e] If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

[f] If type 1 protection is enabled, then the device server shall check the logical block reference tag by comparing it to the lower four bytes of the LBA associated with the logical block. If type 3 protection is enabled, then the device server may check each logical block reference tag if the ATO bit is set to one in the Control mode page (see SPC-4), and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG. The method for acquiring this knowledge is not defined by this standard.

### 5.6 ORWRITE (32) command

The ORWRITE (32) command (see table 48) requests that the device server perform one of the following ORWRITE command (see 4.29) operations:

a) a change generation and clear operation (see 4.29.3); or
b) a set operation (see 4.29.4).

**Table 48 — ORWRITE (32) command**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (7Fh) | | | | | | | |
| 1 | CONTROL | | | | | | | |
| 2 | Reserved | | | | | BMOP | | |
| 3 | Reserved | | | | PREVIOUS GENERATION PROCESSING | | | |
| 4 | Reserved | | | | | | | |
| 5 | | | | | | | | |
| 6 | Reserved | | | GROUP NUMBER | | | | |
| 7 | ADDITIONAL CDB LENGTH (18h) | | | | | | | |
| 8 | (MSB) | | | | | | | |
| 9 | SERVICE ACTION (000Eh) | | | | | | | (LSB) |
| 10 | ORPROTECT | | | DPO | FUA | Reserved | FUA_NV | Reserved |
| 11 | Reserved | | | | | | | |
| 12 | (MSB) | | | | | | | |
| ••• | LOGICAL BLOCK ADDRESS | | | | | | | |
| 19 | | | | | | | | (LSB) |
| 20 | (MSB) | | | | | | | |
| ••• | EXPECTED ORWGENERATION | | | | | | | |
| 23 | | | | | | | | (LSB) |
| 24 | (MSB) | | | | | | | |
| ••• | NEW ORWGENERATION | | | | | | | |
| 27 | | | | | | | | (LSB) |
| 28 | (MSB) | | | | | | | |
| ••• | TRANSFER LENGTH | | | | | | | |
| 31 | | | | | | | | (LSB) |

The OPERATION CODE byte, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-4 and shall be set to the values shown in table 45 for the ORWRITE (16) command.

The CONTROL byte is defined in SAM-5.

The bitmap operation (BMOP) field specifies the operation as described in table 49.

**Table 49 — BMOP field**

| Code | Description |
|---|---|
| 000b | The device server shall process a set operation (see 4.29.4), and the contents of the PREVIOUS GENERATION PROCESSING field and NEW ORWGENERATION field shall be ignored. |
| 001b | The device server shall process a change generation and clear operation (see 4.29.3). |
| All others | Reserved |

The PREVIOUS GENERATION PROCESSING field specifies the policy for processing future set operations that is to be established in the device server by a successful change generation and clear operation (see 4.29.2.2).

See the ORWRITE (16) command (see 5.5) for the definitions of the FUA bit, the FUA_NV bit, the DPO bit, the LOGICAL BLOCK ADDRESS field, and the GROUP NUMBER field.

The EXPECTED ORWGENERATION field contains a code that is compared with generation codes established and maintained by the device server.

The NEW ORWGENERATION field specifies the current ORWgeneration code that is to be established in the device server by a successful change generation and clear operation (see 4.29.3).

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that are read, transferred from the Data-Out Buffer, and processed by the device server using a bitmap buffer (see 3.2.9), starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. If the specified LBA and the specified transfer length exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.6.3).

The device server shall:

a) check protection information read from the medium based on the ORPROTECT field as described in table 46; and
b) check protection information transferred from the Data-Out Buffer based on the ORPROTECT field as described in table 47.

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

## 5.7 POPULATE TOKEN command

### 5.7.1 POPULATE TOKEN command overview

The POPULATE TOKEN command (see table 50) requests that the copy manager (see SPC-4) create a point in time ROD token that represents the specified logical blocks (see 4.30).

Each logical block represented by the point in time ROD token includes user data and may include protection information.

**Table 50 — POPULATE TOKEN command**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (83h) | | | | | | | |
| 1 | Reserved | | | SERVICE ACTION (10h) | | | | |
| 2 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 5 | | | | | | | | |
| 6 | (MSB) | | | LIST IDENTIFIER | | | | |
| ••• | | | | | | | | |
| 9 | | | | | | | | (LSB) |
| 10 | (MSB) | | | PARAMETER LIST LENGTH | | | | |
| ••• | | | | | | | | |
| 13 | | | | | | | | (LSB) |
| 14 | Reserved | | | GROUP NUMBER | | | | |
| 15 | CONTROL | | | | | | | |

The OPERATION CODE byte and the SERVICE ACTION field are defined in SPC-4 and shall be set to the values shown in table 50 for the POPULATE TOKEN command.

The LIST IDENTIFIER field is defined in SPC-4. The list identifier shall be processed as if the LIST ID USAGE field in the parameter data for an EXTENDED COPY(LID4) command (see SPC-4) is set to 00b.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter list that is available to be transferred from the Data-Out Buffer.If the parameter list length is greater than zero and less than 0010h (i.e., 16), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to PARAMETER LIST LENGTH ERROR. A PARAMETER LIST LENGTH field set to zero specifies that no data shall be sent. This shall not be considered an error.

See the PRE-FETCH (10) command (see 5.8) and 4.23 for the definition of the GROUP NUMBER field.

The CONTROL byte is defined in SAM-5.

### 5.7.2 POPULATE TOKEN parameter list

The parameter list for the POPULATE TOKEN command is shown in table 51.

**Table 51 — POPULATE TOKEN parameter list**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | POPULATE TOKEN DATA LENGTH (n - 1) | | | | |
| 1 | | | | | | | | (LSB) |
| 2 | Reserved | | | | | | RTV | IMMED |
| 3 | Reserved | | | | | | | |
| 4 | (MSB) | | | INACTIVITY TIMEOUT | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | (MSB) | | | ROD TYPE | | | | |
| ••• | | | | | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | | | | Reserved | | | | |
| 13 | | | | | | | | |
| 14 | (MSB) | | | BLOCK DEVICE RANGE DESCRIPTOR LENGTH (n - 15) | | | | |
| 15 | | | | | | | | (LSB) |
| **Block device range descriptor list** | | | | | | | | |
| 16 | | | | Block device range descriptor [first] (see 5.7.3) | | | | |
| ••• | | | | | | | | |
| 31 | | | | | | | | |
| | | | | ⋮ | | | | |
| n - 15 | | | | Block device range descriptor [last] (see 5.7.3) | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | |

The POPULATE TOKEN DATA LENGTH field specifies the length in bytes of the data that is available to be transferred from the Data-Out Buffer. The populate token data length does not include the number of bytes in the POPULATE TOKEN DATA LENGTH field. If the POPULATE TOKEN DATA LENGTH field is less than 001Eh (i.e., 30), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

A ROD type valid (RTV) bit set to zero specifies that the copy manager may create a ROD token with any point in time copy ROD type and shall ignore the contents of the ROD TYPE field. An RTV bit set to one specifies that the copy manager shall use the contents of the ROD TYPE field to create the point in time copy ROD.

An immediate (IMMED) bit set to zero specifies that the copy manager shall return status after the specified operations have completed. An IMMED bit set to one specifies that the copy manager shall return status after the entire parameter list has been transferred.

If the INACTIVITY TIMEOUT field is not set to zero, then the INACTIVITY TIMEOUT field specifies the number of seconds that the copy manager should wait for the next third-party copy command (see SPC-4) that uses the ROD token created by the processing of this command before the copy manager invalidates that ROD token due to expiration of this timeout. If the INACTIVITY TIMEOUT field is set to a value larger than the value in the MAXIMUM INACTIVITY TIMEOUT field in the Block Device ROD Token Limits descriptor (see 6.6.6.3), then the

copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the INACTIVITY TIMEOUT field is set to zero, then the DEFAULT INACTIVITY TIMEOUT field in the Block Device ROD Token Limits descriptor (see 6.6.6.3) specifies the number of seconds that the copy manager should wait for the next third-party copy command (see SPC-4) that uses the ROD token created by the processing of this command before the copy manager invalidates that ROD token due to expiration of this timeout.

If the RTV bit is set to one, then the ROD TYPE field specifies the ROD type (see SPC-4) for creating the point in time copy ROD token. The copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST, if:

a) the copy manager does not support the specified ROD type for use with the POPULATE TOKEN command; or
b) the ROD TYPE field specifies a ROD type (see SPC-4) that is not a point in time copy ROD.

The BLOCK DEVICE RANGE DESCRIPTOR LENGTH field specifies the length in bytes of the block device range descriptor list. The block device range descriptor list length should be a multiple of 16. If the block device range descriptor list length is not a multiple of 16, then the last block device range descriptor is incomplete and shall be ignored. If the block device range descriptor list length is less than 16, then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If any block device range descriptors in the block device range descriptor list are truncated due to the parameter list length in the CDB, then those block device range descriptors shall be ignored.

If the number of complete block device range descriptors is larger than the maximum range descriptors value in the Block Device ROD Token Limits descriptor, then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to TOO MANY SEGMENT DESCRIPTORS.

If the same LBA is included in more than one block device range descriptor, then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the number of bytes of user data represented by the sum of the contents of the NUMBER OF LOGICAL BLOCKS fields in all of the complete block device range descriptors is larger than the maximum bytes in block ROD value in the Block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page (see SPC-4), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. If the maximum bytes in block ROD value in the Block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page is not reported, then the maximum token transfer size value in the Block Device ROD Token Limits descriptor (see 6.6.6.2) may indicate a different value for the maximum.

### 5.7.3 Block device range descriptor

The block device range descriptor is described in table 52.

**Table 52 — Block device range descriptor**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | | | | LOGICAL BLOCK ADDRESS | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | (MSB) | | | | | | | |
| ••• | | | | NUMBER OF LOGICAL BLOCKS | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | | | | | | | | |
| ••• | | | | Reserved | | | | |
| 15 | | | | | | | | |

The LOGICAL BLOCK ADDRESS field specifies the first LBA on which the copy manager shall operate for this block device range descriptor.

The NUMBER OF LOGICAL BLOCKS field specifies the number of logical blocks on which the copy manager shall operate for this block device range descriptor beginning with the LBA specified by the LOGICAL BLOCK ADDRESS field.

Processing of block device range descriptors with a number of logical blocks that is not a multiple of the OPTIMAL BLOCK ROD LENGTH GRANULARITY field in the Block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page (see SPC-4) may incur significant delays in processing. If the OPTIMAL BLOCK ROD LENGTH GRANULARITY field in the Block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page is not reported, then the optimal transfer length granularity in the Block Limits VPD page (see 6.6.3) may indicate a different value for the granularity.

For a POPULATE TOKEN command, processing of block device range descriptors where the number of bytes of user data contained in the number of logical blocks exceeds the OPTIMAL BYTES TO TOKEN PER SEGMENT field in the Block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page may incur significant delays in processing.

For a WRITE USING TOKEN command, processing of block device range descriptors where the number of bytes of user data contained in the number of logical blocks exceeds the OPTIMAL BYTES FROM TOKEN PER SEGMENT field in the Block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page may incur significant delays in processing.

If the number of bytes of user data contained in the number of logical blocks is greater than the value in the MAXIMUM BYTES IN BLOCK ROD field in the ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page, then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. If the MAXIMUM BYTES IN BLOCK ROD field in the Block ROD device type specific features descriptor is not reported, then the maximum transfer length in the Block Limits VPD page may indicate a different value for the maximum.

If the NUMBER OF LOGICAL BLOCKS field is set to zero, then the copy manager shall perform no operation for this block device range descriptor. This condition shall not be considered an error.

If the specified LBA and the specified number of logical blocks exceed the capacity of the medium (see 4.5), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

### 5.8 PRE-FETCH (10) command

The PRE-FETCH (10) command (see table 53) requests that the device server:

a) for any mapped LBAs specified by the command, if required by the rules for caching (see 4.15), perform read medium operations on the logical blocks referenced by the LBAs specified by this command; and
b) for any unmapped LBAs specified by the command, update the volatile cache and/or non-volatile cache to prevent retrieval of stale data.

Logical blocks include user data and, if the medium is formatted with protection information enabled, protection information. No data shall be transferred to the Data-In Buffer.

NOTE 8 - Migration from the PRE-FETCH (10) command to the PRE-FETCH (16) command is recommended for all implementations.

**Table 53 — PRE-FETCH (10) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 53 for the PRE-FETCH (10) command.

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

The LOGICAL BLOCK ADDRESS field specifies the LBA of the first logical block (see 4.5) accessed by this command.

The GROUP NUMBER field specifies the group into which attributes associated with the command should be collected (see 4.23). A GROUP NUMBER field set to zero specifies that any attributes associated with the command shall not be collected into any group.

The PREFETCH LENGTH field specifies the number of contiguous logical blocks that shall be pre-fetched (i.e., transferred to the cache from the medium), starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. A PREFETCH LENGTH field set to zero specifies that all logical blocks starting with the one referenced by the LBA specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium shall be pre-fetched. Any other value specifies the number of logical blocks that shall be pre-fetched. If the specified LBA and the specified prefetch length exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The device server is not required to transfer logical blocks that already are contained in the cache.

The CONTROL byte is defined in SAM-5.

If the IMMED bit is set to zero, and the specified logical blocks were transferred to the cache without error, then the device server shall complete the command with CONDITION MET status.

If the IMMED bit is set to zero and the cache does not have sufficient capacity to accept all of the specified logical blocks, then the device server shall transfer to the cache as many of the specified logical blocks that fit. If these logical blocks are transferred without error, then the device server shall complete the command with GOOD status.

If the IMMED bit is set to one and the cache has sufficient capacity to accept all of the specified logical blocks, then the device server shall complete the command with CONDITION MET status.

If the IMMED bit is set to one and the cache does not have sufficient capacity to accept all of the specified logical blocks, then the device server shall complete the command with GOOD status.

If the IMMED bit is set to zero and one or more of the specified logical blocks were not successfully transferred to the cache for reasons other than lack of cache capacity, then the device server shall terminate the command with CHECK CONDITION status with the sense key and the additional sense code set to the appropriate values. If the IMMED bit is set to one and one or more of the specified logical blocks were not successfully transferred to the cache for reasons other than lack of cache capacity, the device server shall report a deferred error (see SPC-4).

### 5.9 PRE-FETCH (16) command

The PRE-FETCH (16) command (see table 54) requests that the device server perform the actions defined for the PRE-FETCH (10) command (see 5.8).

Logical blocks include user data and, if the medium is formatted with protection information enabled, protection information. No data shall be transferred to the Data-In Buffer.

**Table 54 — PRE-FETCH (16) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 54 for the PRE-FETCH (16) command.

See the PRE-FETCH (10) command (see 5.8) for the definitions of the other fields in this command.

### 5.10 PREVENT ALLOW MEDIUM REMOVAL command

The PREVENT ALLOW MEDIUM REMOVAL command (see table 55) requests that the logical unit enable or disable the removal of the medium. If medium removal is prevented on any I_T nexus that has access to the logical unit, then the logical unit shall not allow medium removal.

**Table 55 — PREVENT ALLOW MEDIUM REMOVAL command**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (1Eh) | | | | | | | |
| 1 | Reserved | | | | | | | |
| 2 | Reserved | | | | | | | |
| 3 | Reserved | | | | | | | |
| 4 | Reserved | | | | | | | PREVENT |
| 5 | CONTROL | | | | | | | |

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 55 for the PREVENT ALLOW MEDIUM REMOVAL command.

Table 56 defines the PREVENT field values and their meanings.

**Table 56 — PREVENT field**

| Value | Description |
|---|---|
| 00b | Medium removal shall be allowed. |
| 01b | Medium removal shall be prohibited. |
| 10b to 11b | Obsolete |

The CONTROL byte is defined in SAM-5.

The prevention of medium removal shall begin when any application client issues a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field set to 01b (i.e., medium removal prevented). The prevention of medium removal for the logical unit shall terminate after:

a) one of the following occurs for each I_T nexus through which medium removal had been prevented:
   A) receipt of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field set to 00b; or
   B) an I_T nexus loss;
b) a power on;
c) a hard reset; or
d) a logical unit reset.

If possible, the device server shall perform a synchronize cache operation before terminating the prevention of medium removal.

If a persistent reservation or registration is being preempted by a PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action (see SPC-4), then the equivalent of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field set to zero shall be processed for each the I_T nexuses associated with the persistent reservation or registrations being preempted. This allows an application client to override the prevention of medium removal function for an initiator port that is no longer operating correctly.

While a prevention of medium removal condition is in effect, the logical unit shall inhibit mechanisms that allow removal of the medium by an operator.

### 5.11 READ (10) command

The READ (10) command (see table 57) requests that the device server:

a) if required by the rules for caching (see 4.15), perform read medium operations on the logical blocks referenced by the LBA(s) specified by this command: and
b) transfer the requested logical block data to the Data-In Buffer.

The logical block data transferred to the Data-In Buffer shall include protection information based on the value in the RDPROTECT field (see table 58) and the medium format.

NOTE 9 - Migration from the READ (10) command to the READ (16) command is recommended for all implementations.

**Table 57 — READ (10) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 57 for the READ (10) command.

The device server shall check the protection information read from the medium before returning status for the command based on the RDPROTECT field as described in table 58. All footnotes for table 58 are at the end of the table.

**Table 58 — RDPROTECT field** (part 1 of 3)

| Code | Logical unit formatted with protection information | Shall device server transmit protection information? | Field in protection information [h] | Extended INQUIRY Data VPD page bit value [g] | If check fails [d] [f], additional sense code |
|---|---|---|---|---|---|
| 000b | Yes [j] | No | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | | GRD_CHK = 0 | No check performed |
| | | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | | APP_CHK = 0 | No check performed |
| | | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [i] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | | REF_CHK = 0 | No check performed |
| | No | | No protection information available to check | | |
| 001b<br>101b [b] | Yes | Yes [e] | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | | GRD_CHK = 0 | No check performed |
| | | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | | APP_CHK = 0 | No check performed |
| | | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [i] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | | REF_CHK = 0 | No check performed |
| | No [a] | No protection information available to transmit to the Data-In Buffer or for checking | | | |

**Table 58 — RDPROTECT field** (part 2 of 3)

| Code | Logical unit formatted with protection information | Shall device server transmit protection information? | Field in protection information [h] | Extended INQUIRY Data VPD page bit value [g] | If check fails [d f], additional sense code |
|---|---|---|---|---|---|
| 010b [b] | Yes | Yes [e] | LOGICAL BLOCK GUARD | No check performed | |
| | | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | | APP_CHK = 0 | No check performed |
| | | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [i] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | | REF_CHK = 0 | No check performed |
| | No [a] | No protection information available to transmit to the Data-In Buffer or for checking | | | |
| 011b [b] | Yes | Yes [e] | LOGICAL BLOCK GUARD | No check performed | |
| | | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No [a] | No protection information available to transmit to the Data-In Buffer or for checking | | | |
| 100b [b] | Yes | Yes [e] | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | | GRD_CHK = 0 | No check performed |
| | | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No [a] | No protection information available to transmit to the Data-In Buffer or for checking | | | |
| 110b to 111b | Reserved | | | | |

**Table 58 — RDPROTECT field** (part 3 of 3)

| Code | Logical unit formatted with protection information | Shall device server transmit protection information? | Field in protection information [h] | Extended INQUIRY Data VPD page bit value [g] | If check fails [d f], additional sense code |
|---|---|---|---|---|---|

[a] If a logical unit supports protection information (see 4.22) but has not been formatted with protection information, then the device server shall terminate a command specifying a verify medium operation to the logical unit with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

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

[c] If the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field, then the device server shall check each logical block application tag. If the ATO bit in the Control mode page (see SPC-4) is set to one, then this knowledge is acquired from:

a) the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB, if a READ (32) command (see 5.14) is received by the device server;
b) the Application Tag mode page (see 6.5.3), if a command other than READ (32) is received by the device server, and the ATMPE bit in the Control mode page (see SPC-4) is set to one; or
c) a method not defined by this standard, if a command other than READ (32) is received by the device server, and the ATMPE bit is set to zero.

[d] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

[e] The device server shall transmit protection information to the Data-In Buffer.

[f] If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

[g] See the Extended INQUIRY Data VPD page (see SPC-4) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bit.

[h] If the device server detects:

a) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, and type 1 protection (see 4.22.2.3) or type 2 protection (see 4.22.2.4) is enabled; or
b) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, the LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.22.2.5) is enabled,

then the device server shall not check any protection information in the associated protection information interval.

[i] If type 1 protection is enabled, then the device server shall check the logical block reference tag by comparing it to the lower four bytes of the LBA associated with the logical block. If type 2 protection or type 3 protection is enabled, and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server shall check each logical block reference tag. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a READ (32) command (see 5.14). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

[j] If the DPICZ bit in the Control mode page (see SPC-4) is set to one, then protection information shall not be checked.

A disable page out (DPO) bit set to zero specifies that the retention priority shall be determined by the RETENTION PRIORITY fields in the Caching mode page (see 6.5.5). A DPO bit set to one specifies that the device server shall assign the logical blocks accessed by this command the lowest retention priority for being fetched into or retained by the cache. A DPO bit set to one overrides any retention priority specified in the Caching mode page. All other aspects of the algorithm implementing the cache replacement strategy are not defined by this standard.

NOTE 10 - The DPO bit is used to control replacement of logical blocks in the cache when the application client has information on the future usage of the logical blocks. If the DPO bit is set to one, then the application client is specifying that the logical blocks accessed by the command are not likely to be accessed again in the

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

The force unit access (FUA) bit and the force unit access non-volatile cache (FUA_NV) bit are defined in table 59.

**Table 59 — Force unit access for reads**

| FUA | FUA_NV | Description |
|---|---|---|
| 0 | 0 | The device server may read the logical blocks from volatile cache, non-volatile cache, and/or the medium. |
| 0 | 1 | If the NV_SUP bit is set to one in the Extended INQUIRY Data VPD page (see SPC-4), then the device server shall read the logical blocks from non-volatile cache or the medium. If a non-volatile cache is present and a volatile cache contains a more recent version of a logical block, then the device server shall write the logical block to:<br>a) non-volatile cache; and/or<br>b) the medium,<br>before reading it.<br><br>If the NV_SUP bit is set to zero in the Extended INQUIRY Data VPD page (see SPC-4), then the device server may read the logical blocks from volatile cache, non-volatile cache, and/or the medium. |
| 1 | 0 or 1 | The device server shall perform a read medium operation. If a cache contains a more recent version of a logical block, then the device server shall perform a write medium operation on that logical block before performing a read medium operation on that logical block. |

If rebuild assist mode (see 4.20) is supported and not enabled, then the device server shall ignore the rebuild assist recovery control (RARC) bit. If rebuild assist mode is supported and enabled, then:

a) an RARC bit set to one specifies that the device server shall process reads as defined in 4.20.3.2 and 4.20.3.3; and
b) an RARC bit set to zero is ignored.

If the rebuild assist mode is not supported and the RARC bit is set to one, then the device server should terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

See the PRE-FETCH (10) command (see 5.8) for the definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command and 4.23 for the definition of the GROUP NUMBER field.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be read and transferred to the Data-In Buffer, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. A TRANSFER LENGTH field set to zero specifies that no logical blocks shall be read or transferred. This condition shall not be considered an error. Any other value specifies the number of logical blocks that shall be read and transferred. If the specified LBA and the specified transfer length exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.6.3).

The CONTROL byte is defined in SAM-5.

### 5.12 READ (12) command

The READ (12) command (see table 60) requests that the device server perform the actions defined for the READ (10) command (see 5.11).

NOTE 11 - Migration from the READ (12) command to the READ (16) command is recommended for all implementations.

**Table 60 — READ (12) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 60 for the READ (12) command.

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

### 5.13 READ (16) command

The READ (16) command (see table 61) requests that the device server perform the actions defined for the READ (10) command (see 5.11).

**Table 61 — READ (16) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 61 for the READ (16) command.

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

### 5.14 READ (32) command

The READ (32) command (see table 62) requests that the device server perform the actions defined for the READ (10) command (see 5.11).

The device server shall only process a READ (32) command if type 2 protection is enabled (see 4.22.2.4).

**Table 62 — READ (32) command**

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

The OPERATION CODE byte, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-4 and shall be set to the values shown in table 62 for the READ (32) command.

See the READ (10) command (see 5.11) for the definitions of the CONTROL byte, the GROUP NUMBER field, the RDPROTECT field, the DPO bit, the FUA bit, the RARC bit, the FUA_NV bit, the LOGICAL BLOCK ADDRESS field, and the TRANSFER LENGTH field.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 58 in 5.11), the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.22.3).

If the ATO bit is set to one in the Control mode page (see SPC-4), and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 58 in 5.11), then the LOGICAL BLOCK APPLICATION TAG MASK field contains

a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK field bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information.

The LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored if:

a) the ATO bit is set to zero; or
b) the ATO bit is set to one in the Control mode page (see SPC-4), and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 58 in 5.11).

## 5.15 READ CAPACITY (10) command

### 5.15.1 READ CAPACITY (10) overview

The READ CAPACITY (10) command (see table 63) requests that the device server transfer eight bytes of parameter data describing the capacity and medium format of the direct-access block device to the Data-In Buffer. This command may be processed as if it has a HEAD OF QUEUE task attribute (see 4.16). If the logical unit supports protection information (see 4.22) or logical block provisioning management (see 4.7), then the application client should use the READ CAPACITY (16) command (see 5.16) instead of the READ CAPACITY (10) command.

NOTE 12 - Migration from the READ CAPACITY (10) command to the READ CAPACITY (16) command is recommended for all implementations.

**Table 63 — READ CAPACITY (10) command**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (25h) | | | | | | | |
| 1 | Reserved | | | | | | | Obsolete |
| 2 | Obsolete | | | | | | | |
| ••• | | | | | | | | |
| 5 | | | | | | | | |
| 6 | Reserved | | | | | | | |
| 7 | | | | | | | | |
| 8 | Reserved | | | | | | | Obsolete |
| 9 | CONTROL | | | | | | | |

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 63 for the READ CAPACITY (10) command.

The CONTROL byte is defined in SAM-5.

### 5.15.2 READ CAPACITY (10) parameter data

The READ CAPACITY (10) parameter data is defined in table 64. Any time the READ CAPACITY (10) parameter data changes, the device server should establish a unit attention condition as described in 4.10.

This command may be processed as if it has a HEAD OF QUEUE task attribute (see 4.16).

**Table 64 — READ CAPACITY (10) parameter data**

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

The device server shall set the RETURNED LOGICAL BLOCK ADDRESS field to the lower of:

a) the LBA of the last logical block on the direct-access block device; or
b) FFFF_FFFFh, if the LBA of the last logical block on the direct-access block device is greater than the maximum value that is able to be specified in the RETURNED LOGICAL BLOCK ADDRESS field.

If the RETURNED LOGICAL BLOCK ADDRESS field is set to FFFF_FFFFh, then the application client should issue a READ CAPACITY (16) command (see 5.16) to request that the device server transfer the READ CAPACITY (16) parameter data to the Data-In Buffer.

The LOGICAL BLOCK LENGTH IN BYTES field contains the number of bytes of user data in the logical block indicated by the RETURNED LOGICAL BLOCK ADDRESS field.

### 5.16 READ CAPACITY (16) command

#### 5.16.1 READ CAPACITY (16) command overview

The READ CAPACITY (16) command (see table 65) requests that the device server transfer parameter data describing the capacity and medium format of the direct-access block device to the Data-In Buffer. This command is mandatory if the logical unit supports protection information (see 4.22) or logical block provisioning management (see 4.7) and is optional otherwise. This command may be processed as if it has a HEAD OF QUEUE task attribute (see 4.16).

**Table 65 — READ CAPACITY (16) command**

| Bit / Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (9Eh) | | | | | | | |
| 1 | Reserved | | | SERVICE ACTION (10h) | | | | |
| 2 | | | | | | | | |
| ••• | | | | Obsolete | | | | |
| 9 | | | | | | | | |
| 10 | (MSB) | | | | | | | |
| ••• | | | | ALLOCATION LENGTH | | | | |
| 13 | | | | | | | | (LSB) |
| 14 | Reserved | | | | | | | Obsolete |
| 15 | CONTROL | | | | | | | |

The OPERATION CODE byte and SERVICE ACTION field are defined in SPC-4 and shall be set to the values shown in table 65 for the READ CAPACITY (16) command.

The ALLOCATION LENGTH field is defined in SPC-4.

The CONTROL byte is defined in SAM-5.

### 5.16.2 READ CAPACITY (16) parameter data

The READ CAPACITY (16) parameter data is defined in table 66. Any time the READ CAPACITY (16) parameter data changes, the device server should establish a unit attention condition as described in 4.10.

**Table 66 — READ CAPACITY (16) parameter data**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | RETURNED LOGICAL BLOCK ADDRESS | | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | (MSB) | | | | | | | |
| ••• | LOGICAL BLOCK LENGTH IN BYTES | | | | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | Reserved | | | | | P_TYPE | | PROT_EN |
| 13 | P_I_EXPONENT | | | | | LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT | | |
| 14 | LBPME | LBPRZ | (MSB) | | | | | |
| 15 | LOWEST ALIGNED LOGICAL BLOCK ADDRESS | | | | | | | (LSB) |
| 16 | | | | | | | | |
| ••• | Reserved | | | | | | | |
| 31 | | | | | | | | |

The RETURNED LOGICAL BLOCK ADDRESS field and LOGICAL BLOCK LENGTH IN BYTES field of the READ CAPACITY (16) parameter data are the same as in the READ CAPACITY (10) parameter data (see 5.15). The maximum value that shall be returned in the RETURNED LOGICAL BLOCK ADDRESS field is FFFF_FFFF_FFFF_FFFEh.

The protection type (P_TYPE) field and the protection enable (PROT_EN) bit (see table 67) indicate the logical unit's current type of protection.

**Table 67 — P_TYPE field and PROT_EN bit**

| P_TYPE | PROT_EN | Description |
|---|---|---|
| 000b to 111b | 0 | The logical unit is formatted to type 0 protection (see 4.22.2.2). |
| 000b | 1 | The logical unit is formatted to type 1 protection (see 4.22.2.3). |
| 001b | | The logical unit is formatted to type 2 protection (see 4.22.2.4). |
| 010b | | The logical unit is formatted to type 3 protection (see 4.22.2.5). |
| 011b to 111b | | Reserved |

The P_I_EXPONENT field may be used to determine the number of protection information intervals placed within each logical block (see 5.3.2).

The number of protection information intervals is calculated as follows:

$$\text{number of protection information intervals} = 2^{(\text{p\_i exponent})}$$

where:

p_i exponent is the contents of the P_I EXPONENT field

The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field is defined in table 68.

**Table 68 — LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field**

| Code | Description |
|---|---|
| 0 | One or more physical blocks per logical block [a] |
| n > 0 | $2^n$ logical blocks per physical block |
| [a] The number of physical blocks per logical block is not reported. | |

A logical block provisioning management enabled (LBPME) bit set to one indicates that the logical unit implements logical block provisioning management (i.e., is resource provisioned or thin provisioned) (see 4.7.3). An LBPME bit set to zero indicates that the logical unit does not implement logical block provisioning management (i.e., is fully provisioned (see 4.7.2)).

A logical block provisioning read zeros (LBPRZ) bit set to one indicates that, for read commands specifying an unmapped LBA (see 4.7.4.2), the device server returns user data set to zero and protection information, if any, set to FFFF_FFFF_FFFF_FFFFh. An LBPRZ bit set to zero indicates that, for read commands specifying an unmapped LBA, the device server returns user data set to vendor specific data and protection information, if any, set to FFFF_FFFF_FFFF_FFFFh.

The LOWEST ALIGNED LOGICAL BLOCK ADDRESS field indicates the LBA of the first logical block that is located at the beginning of a physical block (see 4.6).

NOTE 13 - The highest value (i.e., LBA) that the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field supports is 3FFFh (i.e., 16 383).

## 5.17 READ DEFECT DATA (10) command

### 5.17.1 READ DEFECT DATA (10) command overview

The READ DEFECT DATA (10) command (see table 69) requests that the device server transfer parameter data (see 5.17.2) containing a four-byte header, the PLIST, and/or the GLIST to the Data-In Buffer.

If the device server is unable to access a specified defect list due to a medium error, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to DEFECT LIST NOT FOUND.

If the device server is unable to access a specified defect list due to an error other than a medium error or because a specified defect list does not exist, then the device server shall either:

1) terminate the command with CHECK CONDITION status with the sense key set to NO SENSE and the additional sense code set to DEFECT LIST NOT FOUND; or
2) return only the READ DEFECT DATA parameter data header, with the DEFECT LIST LENGTH field set to 0000h.

NOTE 14 - Some device servers are not able to return a defect list until after a FORMAT UNIT command (see 5.2) has been completed without error.

NOTE 15 - Migration from the READ DEFECT DATA (10) command to the READ DEFECT DATA (12) command is recommended for all implementations.

**Table 69 — READ DEFECT DATA (10) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 69 for the READ DEFECT DATA (10) command.

Table 70 defines the request PLIST (REQ_PLIST) bit and the request GLIST (REQ_GLIST) bit.

**Table 70 — REQ_PLIST bit and REQ_GLIST bit**

| REQ_PLIST | REQ_GLIST | Description |
|---|---|---|
| 0 | 0 | The device server shall return only the first four bytes of the READ DEFECT DATA parameter data (i.e., the parameter data header), with the DEFECT LIST LENGTH field set to 0000h. |
| | 1 | The device server shall return the READ DEFECT DATA parameter data header and shall include the GLIST, if any, in the defect list. |
| 1 | 0 | The device server shall return the READ DEFECT DATA parameter data header and shall include the PLIST, if any, in the defect list. |
| | 1 | The device server shall return the READ DEFECT DATA parameter data header and shall include the both the PLIST, if any, and the GLIST, if any, in the defect list. Whether the PLIST and GLIST are merged or not is vendor specific. |

The DEFECT LIST FORMAT field specifies the address descriptor format type (see 6.2) that the device server should use for the defect list. A device server unable to return the requested address descriptor format shall return the address descriptors in their default format and indicate that format type in the DEFECT LIST FORMAT field in the READ DEFECT DATA parameter data header (see 5.17.2 and 5.18.2).

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

The ALLOCATION LENGTH field is defined in SPC-4. If the length of the address descriptors that the device server has to report is greater than the maximum value that is able to be specified by the ALLOCATION LENGTH field, then the device server shall transfer no data and shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The CONTROL byte is defined in SAM-5.

### 5.17.2 READ DEFECT DATA (10) parameter data

The READ DEFECT DATA (10) parameter data (see table 71) contains a four-byte header, followed by zero or more address descriptors.

**Table 71 — READ DEFECT DATA (10) parameter data**

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

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

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

The DEFECT LIST FORMAT field indicates the format of the address descriptors returned in the defect list. This field is defined in 6.2.

If the device server returns short block format address descriptors (see 6.2.2) or long block format address descriptors (see 6.2.5), then the address descriptors contain vendor specific values.

NOTE 16 - The use of the short block format and the long block format is not recommended for this command. There is no standard model that defines the meaning of the block address of a defect. In the usual case, a defect that has been reassigned no longer has an LBA.

The DEFECT LIST LENGTH field indicates the length in bytes of the defect list. The DEFECT LIST LENGTH is equal to four or eight times the number of the address descriptors, depending on the format of the returned address descriptors (see 6.2).

The defect list contains address descriptors (see 6.2).

### 5.18 READ DEFECT DATA (12) command

#### 5.18.1 READ DEFECT DATA (12) command overview

The READ DEFECT DATA (12) command (see table 72) requests that the device server transfer the medium defect data to the Data-In Buffer.

**Table 72 — READ DEFECT DATA (12) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 72 for the READ DEFECT DATA (12) command.

See the READ DEFECT DATA (10) command (see 5.17) for the definitions of the REQ_PLIST bit, the REQ_GLIST bit, the DEFECT LIST FORMAT field, and the CONTROL byte.

NOTE 17 - An application client determines the length of the defect list by sending a READ DEFECT DATA (12) command with an ALLOCATION LENGTH field set to eight. The device server returns the defect list header that contains the length of the defect list.

The ADDRESS DESCRIPTOR INDEX field specifies the index of the first address descriptor (see 6.2) in the defect list that the device server shall return. If the ADDRESS DESCRIPTOR INDEX field is set to:

a) a value less than the number of available address descriptors, then the device server shall transfer a defect list beginning with the address descriptor that is at the ADDRESS DESCRIPTOR INDEX field value multiplied by the size of the address descriptor; or
b) a value greater than or equal to the number of available address descriptors, then the device server shall return a zero length defect list.

The ALLOCATION LENGTH field is defined in SPC-4, except if the length of all the address descriptors that are available is greater than FFFF_FFFFh, then the device server shall transfer the length of address descriptors specified by the allocation length or the DEFECT LIST LENGTH field value plus eight, whichever is less, and complete the command with GOOD status.

### 5.18.2 READ DEFECT DATA (12) parameter data

The READ DEFECT DATA (12) parameter data (see table 73) contains an eight-byte header, followed by zero or more address descriptors.

**Table 73 — READ DEFECT DATA (12) parameter data**

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

The GENERATION CODE field is a two-byte counter that shall be incremented by one by the device server every time the medium defect data is changed. A GENERATION CODE field set to 0000h indicates the generation code is not supported. If the GENERATION CODE field is supported, then the GENERATION CODE field shall be initialized to at least 0001h at power on and the device server shall wrap this field to 0001h as the next increment after reaching its maximum value (i.e., FFFFh).

NOTE 18 - It is recommended that application clients that use the GENERATION CODE field read the field often enough to ensure that it does not increment a multiple of 65 535 times between readings.

The DEFECT LIST LENGTH field indicates the length in bytes of address descriptors from the beginning address descriptor specified by the ADDRESS DESCRIPTOR INDEX field to the last address descriptor available to be returned. A value of FFFF_FFFFh in the DEFECT LIST LENGTH field indicates that more than FFFF_FFFEh bytes are available.

See the READ DEFECT DATA (10) command (see 5.17) for the definitions of the other fields in the READ DEFECT DATA (12) parameter data.

### 5.19 READ LONG (10) command

The READ LONG (10) command (see table 74) requests that the device server transfer data from a single logical block or physical block to the Data-In Buffer. The data transferred during the READ LONG (10) command is vendor specific, but shall include the following items recorded on the medium:

a) if a logical block is being transferred, then:
   A) user data or transformed user data for the logical block;
   B) protection information or transformed protection information, if any, for the logical block; and
   C) any additional information (e.g., ECC bytes) for all the physical blocks in the logical block;
   or
b) if a physical block is being transferred, then:
   A) user data or transformed user data for all the logical blocks in the physical block;
   B) protection information or transformed protection information, if any, for all the logical blocks in the physical block; and
   C) any additional information (e.g., ECC bytes).

If the additional information contain an ECC, then any other additional bytes that are correctable by ECC should be included (e.g., a data synchronization mark within the area covered by ECC). It is not required for the ECC bytes to be at the end of the user data or protection information, if any. However, the ECC bytes should be in the same order as they are on the medium.

If a cache contains a more recent version of the specified logical block or physical block, then the device server shall write the logical block or physical block to the medium before reading it. The values in the Read-Write Error Recovery mode page (see 6.5.8) do not apply to this command. The device server may perform retries while processing this command.

NOTE 19 - Migration from the READ LONG (10) command to the READ LONG (16) command is recommended for all implementations.

**Table 74 — READ LONG (10) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 74 for the READ LONG (10) command.

If there is more than one logical block per physical block (i.e., the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field in the READ CAPACITY (16) parameter data (see 5.16.1) is set to a non-zero value), then:

a) the device server shall support the physical block (PBLOCK) bit;
b) a PBLOCK bit set to one specifies that the device server shall return the entire physical block containing the specified logical block; and
c) a PBLOCK bit set to zero specifies that the device server shall return bytes representing only the specified logical block.

If there are one or more physical blocks per logical block (i.e., the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field in the READ CAPACITY (16) parameter data (see 5.16.1) is set to zero), and the PBLOCK bit is set to one, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

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

The LOGICAL BLOCK ADDRESS field specifies an LBA (see 4.5). If the specified LBA exceeds the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

The BYTE TRANSFER LENGTH field specifies the number of bytes of data that shall be read from the specified logical block or physical block and transferred to the Data-In Buffer. If the BYTE TRANSFER LENGTH field is not set to zero and does not match the available data length, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. In the sense data (see 4.18 and SPC-4), the VALID bit and the ILI bit shall each be set to one, and the INFORMATION field shall be set to the difference (i.e., residue) of the requested byte transfer length minus the actual available data length in bytes. Negative values shall be indicated by two's complement notation.

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

The CONTROL byte is defined in SAM-5.

### 5.20 READ LONG (16) command

The READ LONG (16) command (see table 75) requests that the device server transfer data from a single logical block or physical block to the Data-In Buffer. The data transferred during the READ LONG (16) command is vendor specific, but shall include the following items recorded on the medium:

a) if a logical block is being transferred, then:
   A) user data or transformed user data for the logical block;
   B) protection information or transformed protection information, if any, for the logical block; and
   C) any additional information (e.g., ECC bytes) for all the physical blocks in the logical block;
   or
b) if a physical block is being transferred, then:
   A) user data or transformed user data for all the logical blocks in the physical block;
   B) protection information or transformed protection information, if any, for all the logical blocks in the physical block; and
   C) any additional information (e.g., ECC bytes).

If a cache contains a more recent version of the specified logical block or physical block, then the device server shall write the logical block or physical block to the medium before reading it. The values in the Read-Write Error Recovery mode page (see 6.5.8) do not apply to this command. The device server may perform retries while processing this command. This command is implemented as a service action of the SERVICE ACTION IN operation code (see A.2).

**Table 75 — READ LONG (16) command**

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

The OPERATION CODE byte and SERVICE ACTION field are defined in SPC-4 and shall be set to the values shown in table 75 for the READ LONG (16) command.

See the READ LONG (10) command (see 5.19) for the definitions of the other fields in this command.

## 5.21 REASSIGN BLOCKS command

### 5.21.1 REASSIGN BLOCKS command overview

The REASSIGN BLOCKS command (see table 76) requests that the device server perform a reassignment operation on one or more LBAs (e.g., LBAs referencing logical blocks on which unrecovered read errors occurred) to another area on the medium set aside for this purpose and add the physical blocks containing those logical blocks to the GLIST. This command shall not alter the contents of the PLIST (see 4.13).

The parameter list provided in the Data-Out Buffer contains a reassign LBA list that contains the LBAs of the logical blocks to be reassigned. The device server shall reassign the parts of the medium used for each logical block referenced by an LBA in the reassign LBA list. More than one physical block may be reassigned by each LBA. If the device server recovers logical block data from the original logical block, then the device server shall perform a write medium operation for the recovered logical block data to the logical block referenced by the reassigned LBA.

If the device server does not recover logical block data in a fully provisioned logical unit (see 4.7.2), then the device server shall:

a) write vendor specific data as the user data; and
b) write a default value of FFFF_FFFF_FFFF_FFFFh as the protection information, if enabled (see 4.22.2).

If the device server does not recover logical block data in a resource provisioned logical unit (see 4.7.3.2) or a thin provisioned logical unit (see 4.7.3.3), then the device server shall, for each specified LBA, either:

a) unmap the specified LBA; or
b) perform the following operations:
   A) write vendor specific data as the user data; and
   B) write a default value of FFFF_FFFF_FFFF_FFFFh as the protection information, if enabled.

The vendor specific data written as user data may contain remnants of the original logical block (e.g., partially or fully recovered user data).

The data in all other logical blocks on the medium shall be preserved.

NOTE 20 - The effect of specifying an LBA to be reassigned that previously has been reassigned is to reassign the LBA again.

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

**Table 76 — REASSIGN BLOCKS command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 76 for the REASSIGN BLOCKS command.

A long LBA (LONGLBA) bit set to zero specifies that the reassign LBA list in the REASSIGN BLOCKS parameter list (see 5.21.2) contains four-byte LBAs. A LONGLBA bit set to one specifies that the reassign LBA list in the REASSIGN BLOCKS parameter list contains eight-byte LBAs.

The long list (LONGLIST) bit specifies which parameter list header is used for the REASSIGN BLOCKS parameter list.

The CONTROL byte is defined in SAM-5.

### 5.21.2 REASSIGN BLOCKS parameter list

The REASSIGN BLOCKS parameter list (see table 77) contains a four-byte parameter list header followed by a reassign LBA list containing one or more LBAs.

**Table 77 — REASSIGN BLOCKS parameter list**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Parameter list header (see table 78 or table 79) | | | | | | | |
| ••• | | | | | | | | |
| 3 | | | | | | | | |
| **Reassign LBA list (if any)** | | | | | | | | |
| 4 | Reassign LBA [first] (see table 80 or table 81) | | | | | | | |
| ••• | | | | | | | | |
| 7 or 11 | | | | | | | | |
| n-4 or n-7 | Reassign LBA [last] (see table 80 or table 81) | | | | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | |

If the LONGLIST bit in the REASSIGN BLOCKS command CDB is set to zero, then the parameter list header is defined in table 78.

**Table 78 — REASSIGN BLOCKS short parameter list header**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | | | | | | | |
| 1 | | | | | | | | |
| 2 | (MSB) | REASSIGN LBA LENGTH | | | | | | |
| 3 | | | | | | | | (LSB) |

If the LONGLIST bit in the REASSIGN BLOCKS command CDB is set to one, then the parameter list header is defined in table 79.

**Table 79 — REASSIGN BLOCKS long parameter list header**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | REASSIGN LBA LENGTH | | | | | | | |
| 3 | | | | | | | | (LSB) |

The REASSIGN LBA LENGTH field specifies the total length in bytes of the reassign LBA list. The REASSIGN LBA LENGTH field does not include the parameter list header length and is equal to:

a) four times the number of LBAs, if the LONGLBA in the REASSIGN BLOCKS command CDB bit is set to zero; or
b) eight times the number of LBAs, if the LONGLBA bit is set to one.

The REASSIGN LBA LIST field contains a list of LBAs to be reassigned. The LBAs shall be sorted in ascending order.

If the LONGLBA bit is set to zero, then table 80 defines the reassign LBA.

**Table 80 — Reassign LBA if the LONGLBA bit is set to zero**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | REASSIGN LOGICAL BLOCK ADDRESS | | | | | | | |
| 3 | | | | | | | | (LSB) |

The REASSIGN LOGICAL BLOCK ADDRESS field specifies an LBA to be reassigned.

If the LONGLBA bit is set to one, then table 81 defines the reassign LBA.

**Table 81 — Reassign LBA if the LONGLBA bit is set to one**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | REASSIGN LOGICAL BLOCK ADDRESS | | | | | | | |
| 7 | | | | | | | | (LSB) |

The REASSIGN LOGICAL BLOCK ADDRESS field specifies an LBA to be reassigned.

If a specified LBA exceeds the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE or INVALID FIELD IN PARAMETER LIST. The additional sense code should be set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

If the direct-access block device has insufficient capacity to reassign all of the specified LBAs, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to HARDWARE ERROR and the additional sense code set to NO DEFECT SPARE LOCATION AVAILABLE.

If the direct-access block device is unable to complete a REASSIGN BLOCKS command without error, then the device server shall terminate the command with CHECK CONDITION status with the appropriate sense data (see 4.18 and SPC-4). The first LBA not reassigned shall be returned in the COMMAND-SPECIFIC INFOR-

MATION field of the sense data. If information about the first LBA not reassigned is not available, or if all the LBAs have been reassigned, then the COMMAND-SPECIFIC INFORMATION field shall be set to FFFF_FFFFh if fixed format sense data is being used or FFFF_FFFF_FFFF_FFFFh if descriptor format sense data is being used.

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

NOTE 21 - If the device server terminates the REASSIGN BLOCKS command with CHECK CONDITION status, and the sense data COMMAND-SPECIFIC INFORMATION field contains a valid LBA, then it is recommended that the application client remove all LBAs from the reassign LBA list prior to the one returned in the COMMAND-SPECIFIC INFORMATION field. If the sense key is set to MEDIUM ERROR, and the INFORMATION field contains the valid LBA, then it is recommended that the application client insert that LBA into the reassign LBA list and reissue the REASSIGN BLOCKS command with the new reassign LBA list. Otherwise, it is recommended that the application client perform any corrective action indicated by the sense data and then reissue the REASSIGN BLOCKS command with the new reassign LBA list.

## 5.22 RECEIVE ROD TOKEN INFORMATION

### 5.22.1 RECEIVE ROD TOKEN INFORMATION overview

The RECEIVE ROD TOKEN INFORMATION command (see SPC-4) provides a method for an application client to receive information about the results of a previous or current block device ROD token operation. Table 82 shows the operations and a reference to the subclause where each topic is described.

**Table 82 — RECEIVE ROD TOKEN INFORMATION reference**

| Command originating the operation | Command reference | RECEIVE ROD TOKEN INFORMATION returned parameter data reference |
|---|---|---|
| POPULATE TOKEN | 5.7 | 5.22.2 |
| WRITE USING TOKEN | 5.46 | 5.22.3 |

### 5.22.2 RECEIVE ROD TOKEN INFORMATION parameter data for POPULATE TOKEN command

Table 83 shows the parameter data returned by the copy manager in response to a RECEIVE ROD TOKEN INFORMATION command if the command received on the same I_T nexus with a list identifier that matches the list identifier specified in the RECEIVE ROD TOKEN INFORMATION CDB is a POPULATE TOKEN command.

**Table 83 — RECEIVE ROD TOKEN INFORMATION parameter data for POPULATE TOKEN**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | | | | AVAILABLE DATA (n - 3) | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | | Reserved | | | RESPONSE TO SERVICE ACTION (10h) | | | |
| 5 | Reserved | | | COPY OPERATION STATUS | | | | |
| 6 | (MSB) | | | | | | | |
| 7 | | | | OPERATION COUNTER | | | | (LSB) |
| 8 | (MSB) | | | | | | | |
| ••• | | | | ESTIMATED STATUS UPDATE DELAY | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | | | | EXTENDED COPY COMPLETION STATUS | | | | |
| 13 | | | | LENGTH OF THE SENSE DATA (m - 31) | | | | |
| 14 | | | | SENSE DATA LENGTH | | | | |
| 15 | | | | TRANSFER COUNT UNITS (F1h) | | | | |
| 16 | (MSB) | | | | | | | |
| ••• | | | | TRANSFER COUNT | | | | |
| 23 | | | | | | | | (LSB) |
| 24 | (MSB) | | | | | | | |
| 25 | | | | SEGMENTS PROCESSED (0000h) | | | | (LSB) |
| 26 | | | | | | | | |
| ••• | | | | Reserved | | | | |
| 31 | | | | | | | | |
| 32 | (MSB) | | | | | | | |
| ••• | | | | SENSE DATA | | | | |
| m | | | | | | | | (LSB) |
| m + 1 | (MSB) | | | | | | | |
| ••• | | | | ROD TOKEN DESCRIPTOR LENGTH (n - (m + 4)) | | | | |
| m + 4 | | | | | | | | (LSB) |
| m + 5 | | | | | | | | |
| m + 6 | | | | Restricted (see SPC-4) | | | | |
| m + 7 | | | | | | | | |
| ••• | | | | ROD TOKEN | | | | |
| n | | | | | | | | |

The AVAILABLE DATA field, the COPY OPERATION STATUS field, the OPERATION COUNTER field, the ESTIMATED STATUS UPDATE DELAY field, the EXTENDED COPY COMPLETION STATUS field, the LENGTH OF THE SENSE DATA field, SENSE DATA LENGTH field, the SENSE DATA field, and the ROD TOKEN field are defined in SPC-4.

The RESPONSE TO SERVICE ACTION field is defined in SPC-4 and shall be set to the value shown in table 83 in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a POPULATE TOKEN command.

The TRANSFER COUNT UNITS field is defined in SPC-4 and shall be set to the value shown in table 83 in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a POPULATE TOKEN command.

The TRANSFER COUNT field indicates the number of contiguous logical blocks represented by the ROD token that were read without error from the media starting at the logical block address specified in the first block device range descriptor and including the LBAs described in all block device range descriptors of the POPULATE TOKEN command to which this response applies.

If the value in the TRANSFER COUNT field is not equal to the sum of the contents of the NUMBER OF LOGICAL BLOCKS fields in all of the complete block device range descriptors of the POPULATE TOKEN command to which this response applies, then the COPY OPERATION STATUS field shall be set to 3h. Other values in the COPY OPERATION STATUS field are defined in SPC-4.

The SEGMENTS PROCESSED field is defined in SPC-4 and shall be set to the value shown in table 83 in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a POPULATE TOKEN command.

The ROD TOKEN DESCRIPTOR LENGTH field is defined in SPC-4 and shall be set to the size of the ROD TOKEN field plus two in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a POPULATE TOKEN command.

### 5.22.3 RECEIVE ROD TOKEN INFORMATION parameter data for WRITE USING TOKEN command

Table 84 shows the parameter data returned by the copy manager in response to a RECEIVE ROD TOKEN INFORMATION command if the command received on the same I_T nexus with a list identifier that matches the list identifier specified in the RECEIVE ROD TOKEN INFORMATION CDB is a WRITE USING TOKEN command.

**Table 84 — RECEIVE ROD TOKEN INFORMATION parameter data for WRITE USING TOKEN**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | AVAILABLE DATA (n - 3) | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | Reserved | | | | RESPONSE TO SERVICE ACTION (11h) | | | |
| 5 | Reserved | | | COPY OPERATION STATUS | | | | |
| 6 | (MSB) | | | OPERATION COUNTER | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | (MSB) | | | ESTIMATED STATUS UPDATE DELAY | | | | |
| ••• | | | | | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | EXTENDED COPY COMPLETION STATUS | | | | | | | |
| 13 | LENGTH OF THE SENSE DATA ((n - 4) - 31) | | | | | | | |
| 14 | SENSE DATA LENGTH | | | | | | | |
| 15 | TRANSFER COUNT UNITS (F1h) | | | | | | | |
| 16 | (MSB) | | | TRANSFER COUNT | | | | |
| ••• | | | | | | | | |
| 23 | | | | | | | | (LSB) |
| 24 | (MSB) | | | SEGMENTS PROCESSED (0000h) | | | | |
| 25 | | | | | | | | (LSB) |
| 26 | | | | Reserved | | | | |
| ••• | | | | | | | | |
| 31 | | | | | | | | |
| 32 | (MSB) | | | SENSE DATA | | | | |
| ••• | | | | | | | | |
| n - 4 | | | | | | | | (LSB) |
| n - 3 | | | | Restricted (see SPC-4) | | | | |
| n | | | | | | | | |

The AVAILABLE DATA field, the COPY OPERATION STATUS field, the OPERATION COUNTER field, the ESTIMATED STATUS UPDATE DELAY field, the EXTENDED COPY COMPLETION STATUS field, the LENGTH OF THE SENSE DATA field, the SENSE DATA LENGTH field, and the SENSE DATA field are defined in SPC-4.

The RESPONSE TO SERVICE ACTION field is defined in SPC-4 and shall be set to the value shown in table 84 in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a WRITE USING TOKEN command.

The TRANSFER COUNT UNITS field is defined in SPC-4 and shall be set to the value shown in table 84 in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a WRITE USING TOKEN command.

The TRANSFER COUNT field indicates the number of contiguous logical blocks that were written without error to the medium starting at the logical block address specified in the first block device range descriptor and including the LBAs described in all block device range descriptors of the WRITE USING TOKEN command to which this response applies.

If the value in the TRANSFER COUNT field is not equal to the sum of the contents of the NUMBER OF LOGICAL BLOCKS fields in all of the complete block device range descriptors of the WRITE USING TOKEN command to which this response applies, then the COPY OPERATION STATUS field shall be set to 3h. Other values in the COPY OPERATION STATUS field are defined in SPC-4.

The SEGMENTS PROCESSED field is defined in SPC-4 and shall be set to the value shown in table 84 in response to a RECEIVE ROD TOKEN INFORMATION command in which the LIST IDENTIFIER field specifies a POPULATE TOKEN command.

## 5.23 REPORT REFERRALS command

### 5.23.1 REPORT REFERRALS command overview

The REPORT REFERRALS command requests that the device server send information indicating the user data segment(s) on the logical unit and the SCSI target ports though which those user data segments may be accessed (see 4.28) to the application client. This command shall be supported by a logical unit that reports in the Extended INQUIRY Data VPD page (see SPC-4) that it supports referrals (i.e., the R_SUP bit set to one). This command is implemented as a service action of the SERVICE ACTION IN operation code (see A.2).

**Table 85 — REPORT REFERRALS command**

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

The OPERATION CODE byte and SERVICE ACTION field are defined in SPC-4 and shall be set to the values shown in table 85 for a REPORT REFERRALS command.

The LOGICAL BLOCK ADDRESS field specifies an LBA in the first user data segment the device server shall report in the REPORT REFERRALS parameter data. If the specified LBA exceeds the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

The ALLOCATION LENGTH field is defined in SPC-4.

A one segment (ONE_SEG) bit set to zero specifics that the device server shall return information on all user data segments starting with the user data segment that contains the LBA specified in the LOGICAL BLOCK ADDRESS field and ending with the user data segment that contains the last LBA of the logical unit. A ONE_SEG

bit set to one specifies the device server shall only return information on the user data segment that contains the LBA specified in the LOGICAL BLOCK ADDRESS field.

The CONTROL byte is defined in SAM-5.

### 5.23.2 REPORT REFERRALS parameter data

The REPORT REFERRALS parameter data (see table 86) contains information indicating the user data segment(s) on the logical unit and the SCSI target port groups though which those user data segments may be accessed (see 4.28).

**Table 86 — REPORT REFERRALS parameter data**

| Bit<br>Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | | | | | | | |
| 1 | | | | | | | | |
| 2 | (MSB) | USER DATA SEGMENT REFERRAL DESCRIPTOR LENGTH (y - 3) | | | | | | |
| 3 | | | | | | | | (LSB) |
| **User data segment referral descriptor list** | | | | | | | | |
| 4 | User data segment referral descriptor [first] | | | | | | | |
| ••• | | | | | | | | |
| 4 + n | | | | | | | | |
| | ⋮ | | | | | | | |
| y - m | User data segment referral descriptor [last] | | | | | | | |
| ••• | | | | | | | | |
| y | | | | | | | | |

The USER DATA SEGMENT REFERRAL DESCRIPTOR LENGTH field indicates the number of bytes that follow in the REPORT REFERRALS parameter data.

The user data segment referral descriptor (see table 14) is defined in the user data segment referral sense data descriptor.

## 5.24 SANITIZE command

### 5.24.1 SANITIZE command overview

The SANITIZE command (see table 87) requests that the device server perform a sanitize operation (see 4.11). This device server shall process this command as if it has a HEAD OF QUEUE task attribute (see 4.16).

**Table 87 — SANITIZE command**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (48h) | | | | | | | |
| 1 | IMMED | Reserved | AUSE | SERVICE ACTION | | | | |
| 2 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 6 | | | | | | | | |
| 7 | (MSB) | PARAMETER LIST LENGTH | | | | | | |
| 8 | | | | | | | | LSB) |
| 9 | CONTROL | | | | | | | |

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 87 for a SANITIZE command.

If the immediate (IMMED) bit is set to zero, then the device server shall return status after the sanitize operation is completed. If the IMMED bit set to one, then the device server shall return status as soon as the CDB and parameter data, if any, have been validated. The REQUEST SENSE command may be used to poll for progress of the sanitize operation regardless of the value of the IMMED bit.

If the allow unrestricted sanitize exit (AUSE) bit is set to one, and the specified sanitize operation fails, then the device server shall process a subsequent EXIT FAILURE MODE service action as if the previous sanitize operation had completed without error (see 4.11.3)

If:

a) the AUSE bit is set to zero in the SANITIZE command that requested a sanitize operation;
b) the specified sanitize operation completes with an error; and
c) a subsequent SANITIZE command with the EXIT FAILURE MODE service action is received,

then the device sever shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The SERVICE ACTION field is defined in 5.24.2.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter data that is available to be transferred from the Data-Out Buffer. A PARAMETER LIST LENGTH field set to zero specifies that no data shall be transferred.

The CONTROL byte is defined in SAM-5.

### 5.24.2 SANITIZE command service actions

#### 5.24.2.1 SANITIZE command service actions overview

The SANITIZE command service actions are defined in table 88. At least one service action shall be supported if the SANITIZE command is supported. If the service action specified in the CDB is not supported, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

**Table 88 — SANITIZE service action codes**

| Code | Name | Description | PARAMETER LIST LENGTH requirement [a] | Reference |
|---|---|---|---|---|
| 01h | OVERWRITE | Perform a sanitize overwrite operation | Set to > 0004h and < (logical block length + 5)h | 5.24.2.2 |
| 02h | BLOCK ERASE | Perform a sanitize block erase operation | Set to 0000h | 5.24.2.3 |
| 03h | CRYPTOGRAPHIC ERASE | Perform a sanitize cryptographic erase operation | Set to 0000h | 5.24.2.4 |
| 1Fh | EXIT FAILURE MODE | Exit the sanitize failure mode | Set to 0000h | 5.24.2.5 |
| all others | Reserved | | | |

[a] If the requirement is not met, then the SANITIZE command is terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

#### 5.24.2.2 OVERWRITE service action

The OVERWRITE service action (see table 88) requests that the device server perform a sanitize overwrite operation (see 4.11).

The parameter list format for the OVERWRITE service action is shown in table 89.

**Table 89 — OVERWRITE service action parameter list**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | INVERT | TEST | | OVERWRITE COUNT | | | | |
| 1 | Reserved | | | | | | | |
| 2 | (MSB) | INITIALIZATION PATTERN LENGTH (n - 3) | | | | | | |
| 3 | | | | | | | | LSB) |
| 4 | INITIALIZATION PATTERN | | | | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | |

If the INVERT bit is set to zero, then the initialization pattern and protection information bytes, if any, are written as specified in the INITIALIZATION PATTERN field on each overwrite pass. If the INVERT bit is set to one, then the initialization pattern and protection information bytes, if any, shall be inverted (i.e., each bit XORed with one) between consecutive overwrite passes.

The TEST field is described in table 90.

**Table 90 — TEST field**

| Code | Description |
|---|---|
| 00b | Shall not cause any changes in the defined behavior of the SANITIZE command. |
| 01b to 11b | Vendor specific |

The OVERWRITE COUNT field specifies the number of overwrite passes to be performed. The value of 00h is reserved.

The INITIALIZATION PATTERN LENGTH field specifies the length in bytes of the INITIALIZATION PATTERN field. The INITIALIZATION PATTERN LENGTH field shall be greater than zero and shall not exceed the logical block length. If the INITIALIZATION PATTERN LENGTH field is set to zero or a value greater than the logical block length, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The INITIALIZATION PATTERN field specifies the data pattern to be used to write the user data. This data pattern is repeated as necessary to fill each logical block. For each logical block, the first byte of the user data shall begin with the first byte of the initialization pattern. The protection information, if any, shall be set to FFFF_FFFF_FFFF_FFFFh.

If the INVERT bit is set to one and:

a) the OVERWRITE COUNT field is set to an even number, then the pattern used for the first write pass shall consist of:
   A) the user data set to the inversion of the INITIALIZATION PATTERN data; and
   B) the protection information, if any, set to 0000_0000_0000_0000h;
   or
b) the OVERWRITE COUNT field is set to an odd number, then the pattern used for the first write pass shall consist of:
   A) the user data set to the INITIALIZATION PATTERN data; and
   B) the protection information, if any, set to FFFF_FFFF_FFFF_FFFFh.

After a sanitize overwrite operation completes without error:

a) the medium shall not be format corrupt (see 4.10); and
b) protection information, if any, shall be set to FFFF_FFFF_FFFF_FFFFh in all logical blocks on the medium.

#### 5.24.2.3 BLOCK ERASE service action

The BLOCK ERASE service action (see table 88) requests that the device server perform a sanitize block erase operation (see 4.11).

After a sanitize block erase operation completes without error, the device server may terminate commands that request read medium operations specifying mapped LBAs (see 4.7.1):

a) based on the setting of the WABEREQ field in the Block Device Characteristics VPD page; or
b) if the logical unit is formatted with protection information, then CHECK CONDITION status with the sense key set to ABORTED COMMAND and the additional sense code set to one of the codes defined in table 58.

#### 5.24.2.4 CRYPTOGRAPHIC ERASE service action

The CRYPTOGRAPHIC ERASE service action (see table 88) requests that the device server perform a sanitize cryptographic erase operation (see 4.11).

After a sanitize cryptographic erase operation completes without error, the device server may terminate commands that request read medium operations specifying mapped LBAs (see 4.7.1):

a) based on the setting of the WACEREQ field in the Block Device Characteristics VPD page; or

b) if the logical unit is formatted with protection information, then CHECK CONDITION status with the sense key set to ABORTED COMMAND and the additional sense code set to one of the codes defined in table 58.

#### 5.24.2.5 EXIT FAILURE MODE service action

The EXIT FAILURE MODE service action (see table 88) requests that the device server complete a sanitize operation which completed with an error as if the sanitize operation completed without an error (see 4.11).

After successful completion of a SANITIZE command with the EXIT FAILURE MODE service action:

a) if any LBA is mapped (see 4.7.1), and the logical unit is formatted with protection information, then the device server may terminate read commands to mapped LBAs with CHECK CONDITION status with the sense key set to ABORTED COMMAND and the additional sense code set to one of the codes defined in table 58; and
b) the device server should complete reads to unmapped LBAs without error (see 4.7.4.4.1 and 4.7.4.5.1).

### 5.25 START STOP UNIT command

The START STOP UNIT command (see table 91) requests that the device server change the power condition of the logical unit (see 4.21) or load or eject the medium. This includes specifying that the device server enable or disable the direct-access block device for medium access operations by controlling power conditions and timers.

If any deferred downloaded code has been received as a result of a WRITE BUFFER command with the MODE field set to 0Eh (see SPC-4), then that deferred downloaded code shall replace the current operational code.

**Table 91 — START STOP UNIT command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 91 for a START STOP UNIT command.

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

The combinations of values in the POWER CONDITION field and the POWER CONDITION MODIFIER field are defined in table 92. If the POWER CONDITION field is supported and is set to a value other than 0h, then the device server shall ignore the START bit and the LOEJ bit.

**Table 92 — POWER CONDITION and POWER CONDITION MODIFIER field** (part 1 of 2)

| POWER CONDITION value | POWER CONDITION name | POWER CONDITION MODIFIER value | Device server action |
|---|---|---|---|
| 0h | START_ VALID | 0h | Process the START bit and the LOEJ bit. |
| 1h | ACTIVE | 0h | Cause the logical unit to transition to the active power condition. [a] |
| 2h | IDLE | 0h | Cause the logical unit to transition to the idle_a power condition. [a b] |
| | | 1h | Cause the logical unit to transition to the idle_b power condition. [a b c] |
| | | 2h | Cause the logical unit to transition to the idle_c power condition. [a b d] |
| 3h | STANDBY | 0h | Cause the logical unit to transition to the standby_z power condition. [a b] |
| | | 1h | Cause the logical unit to transition to the standby_y power condition. [a b] |
| 5h | Obsolete | 0h to Fh | Obsolete |
| 7h | LU_ CONTROL | 0h | Initialize and start all of the idle condition timers that are enabled (see SPC-4), and initialize and start all of the standby condition timers that are enabled (see SPC-4). |

[a] Process the following actions:
1) The device server shall comply with any SCSI transport protocol specific power condition transition restrictions (e.g., the NOTIFY (ENABLE SPINUP) requirement (see SPL));
2) the logical unit shall transition to the specified power condition; and
3) the device server shall disable all of the idle condition timers that are enabled (see SPC-4) and disable all of the standby condition timers that are enabled (see SPC-4) until another START STOP UNIT command is processed that returns control of the power condition to the logical unit or a logical unit reset occurs.

[b] If a timer for a background scan operation expires, or a device specific event occurs, then the logical unit shall not leave this power condition to perform the background function associated with the timer or event.

[c] The device server shall cause the direct access block device to increase its tolerance of external physical forces (e.g., causes a device that has movable read/write heads to move those heads to a safe position).

[d] The device server shall cause the direct access block device to increase its tolerance of external physical forces and reduce its power consumption to use less power than when the logical unit is in the idle_b power condition (e.g., cause a device that has rotating medium to rotate the medium at a lower rpm).

[e] If the specified timer is supported and enabled, then the device server shall:
a) force the specified timer to be set to zero, which may cause the logical unit to transition to the specified power condition;
b) initialize and start all of the idle condition timers that are enabled (see SPC-4); and
c) initialize and start all of the standby condition timers that are enabled (see SPC-4),
otherwise the device server shall terminate the START STOP UNIT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

**Table 92 — POWER CONDITION and POWER CONDITION MODIFIER field** (part 2 of 2)

| POWER CONDITION value | POWER CONDITION name | POWER CONDITION MODIFIER value | Device server action |
|---|---|---|---|
| Ah | FORCE_IDLE_0 | 0h | Force the idle_a condition timer to be set to zero. [e] |
| | | 1h | Force the idle_b condition timer to be set to zero. [e] |
| | | 2h | Force the idle_c condition timer to be set to zero. [e] |
| Bh | FORCE_STANDBY_0 | 0h | Force the standby_z condition timer to be set to zero. [e] |
| | | 1h | Force the standby_y condition timer to be set to zero. [e] |
| All other combinations | | | Reserved |

[a] Process the following actions:
1) The device server shall comply with any SCSI transport protocol specific power condition transition restrictions (e.g., the NOTIFY (ENABLE SPINUP) requirement (see SPL));
2) the logical unit shall transition to the specified power condition; and
3) the device server shall disable all of the idle condition timers that are enabled (see SPC-4) and disable all of the standby condition timers that are enabled (see SPC-4) until another START STOP UNIT command is processed that returns control of the power condition to the logical unit or a logical unit reset occurs.

[b] If a timer for a background scan operation expires, or a device specific event occurs, then the logical unit shall not leave this power condition to perform the background function associated with the timer or event.

[c] The device server shall cause the direct access block device to increase its tolerance of external physical forces (e.g., causes a device that has movable read/write heads to move those heads to a safe position).

[d] The device server shall cause the direct access block device to increase its tolerance of external physical forces and reduce its power consumption to use less power than when the logical unit is in the idle_b power condition (e.g., cause a device that has rotating medium to rotate the medium at a lower rpm).

[e] If the specified timer is supported and enabled, then the device server shall:
a) force the specified timer to be set to zero, which may cause the logical unit to transition to the specified power condition;
b) initialize and start all of the idle condition timers that are enabled (see SPC-4); and
c) initialize and start all of the standby condition timers that are enabled (see SPC-4),
otherwise the device server shall terminate the START STOP UNIT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If the START STOP UNIT command specifies a power condition that conflicts with an operation in progress (e.g., a background self test), then, after the START STOP UNIT command completes with GOOD status, the logical unit may not be in the power condition that was requested by the command.

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

If no START STOP UNIT command is being processed by the device server, then the device server shall process any received START STOP UNIT command.

If a START STOP UNIT command is being processed by the device server and a subsequent START STOP UNIT command for which the CDB is validated requests that the logical unit change to the same power condition that was specified by the START STOP UNIT command being processed, then the device server shall process the subsequent command.

If the NO_FLUSH bit is set to zero, then logical units that contain cache shall write all cached logical blocks to the medium (e.g., as they would do in response to a SYNCHRONIZE CACHE command (see 5.26 and 5.27)

with the SYNC_NV bit set to zero, the LOGICAL BLOCK ADDRESS field set to zero, and the NUMBER OF LOGICAL BLOCKS field set to zero) prior to entering into any power condition that prevents accessing the medium (e.g., before the rotating medium spindle motor is stopped during transition to the stopped power condition). If the NO_FLUSH bit is set to one, then cached logical blocks should not be written to the medium by the logical unit prior to entering into any power condition that prevents accessing the medium.

If the load eject (LOEJ) bit is set to zero, then the logical unit shall take no action regarding loading or ejecting the medium. If the LOEJ bit is set to one, then the logical unit shall unload the medium if the START bit is set to zero. If the LOEJ bit is set to one, then the logical unit shall load the medium if the START bit is set to one. If the POWER CONDITION field is supported and is set to a value other than 0h, then the device server shall ignore the LOEJ bit.

If the START bit is set to zero, then the device server shall:

a) cause the logical unit to transition to the stopped power condition;
b) stop any idle condition timer that is enabled (see SPC-4); and
c) stop any standby condition timer that is enabled (see SPC-4).

If the START bit set to one, then the device server shall:

1) comply with requirements defined in SCSI transport protocol standards (e.g., the NOTIFY (ENABLE SPINUP) requirement (see SPL));
2) cause the logical unit to transition to the active power condition;
3) initialize and start any idle condition timer that is enabled; and
4) initialize and start any standby condition timer that is enabled.

If the POWER CONDITION field is supported and is set to a value other than 0h, then the device server shall ignore the START bit.

The CONTROL byte is defined in SAM-5.

### 5.26 SYNCHRONIZE CACHE (10) command

The SYNCHRONIZE CACHE (10) command (see table 93) requests that, depending on the setting of the SYNC_NV bit, the device server either:

a) perform write medium operations on logical blocks referenced by the specified LBAs (i.e., write medium operations are performed on any of the specified logical block data that is in cache and has not already been written to the medium); or
b) write the logical blocks referenced by the specified LBAs to non-volatile cache.

NOTE 22 - Migration from the SYNCHRONIZE CACHE (10) command to the SYNCHRONIZE CACHE (16) command is recommended for all implementations.

**Table 93 — SYNCHRONIZE CACHE (10) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 93 for a SYNCHRONIZE CACHE (10) command.

See the PRE-FETCH (10) command (see 5.8) for the definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command (see 5.8) and 4.23 for the definition of the GROUP NUMBER field.

The SYNC_NV bit (see table 94) specifies whether the device server is required to synchronize volatile and non-volatile caches.

**Table 94 — SYNC_NV bit**

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

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

If the IMMED bit is set to one and the synchronize cache operation has not completed, then the SYNC_PROG field in the Caching mode page (see 6.5.5) defines device server behavior while the synchronize cache command is being processed.

The NUMBER OF LOGICAL BLOCKS field specifies the number of logical blocks that shall be synchronized, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. A NUMBER OF LOGICAL BLOCKS field set to zero specifies that all logical blocks starting with the one referenced by the LBA specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium shall be synchronized. If the specified LBA and the specified number of logical blocks exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

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

The CONTROL byte is defined in SAM-5.

### 5.27 SYNCHRONIZE CACHE (16) command

The SYNCHRONIZE CACHE (16) command (see table 95) requests that the device server perform the actions defined for the SYNCHRONIZE CACHE (10) command (see 5.26).

**Table 95 — SYNCHRONIZE CACHE (16) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 95 for a SYNCHRONIZE CACHE (16) command.

See the SYNCHRONIZE CACHE (10) command (see 5.26) for the definitions of the other fields in this command.

## 5.28 UNMAP command

### 5.28.1 UNMAP command overview

The UNMAP command (see table 96) requests that the device server cause one or more LBAs to be unmapped (see 4.7.3).

**Table 96 — UNMAP command**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (42h) | | | | | | | |
| 1 | Reserved | | | | | | | ANCHOR |
| 2 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 5 | | | | | | | | |
| 6 | Reserved | | | GROUP NUMBER | | | | |
| 7 | (MSB) | PARAMETER LIST LENGTH | | | | | | |
| 8 | | | | | | | | (LSB) |
| 9 | CONTROL | | | | | | | |

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 96 for an UNMAP command.

For a thin provisioned logical unit (see 4.7.3.3):

a) an ANCHOR bit set to zero specifies that any LBA on which an unmap operation (see 4.7.3.4) is performed shall either become deallocated or anchored and should become deallocated; and
b) an ANCHOR bit set to one specifies that any LBA on which an unmap operation is performed shall become anchored.

For a resource provisioned logical unit (see 4.7.3.2), any LBA on which an unmap operation is performed shall become anchored (i.e., the command is processed as if the ANCHOR bit is set to one).

If the ANCHOR bit is set to one, and the ANC_SUP bit in the Logical Block Provisioning VPD page (see 6.6.4) is set to zero, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

See the PRE-FETCH (10) command (see 5.8) and 4.23 for the definition of the GROUP NUMBER field.

The PARAMETER LIST LENGTH field specifies the length in bytes of the UNMAP parameter list that is available to be transferred from the Data-Out Buffer.If the parameter list length is greater than zero and less than 0008h (i.e., eight), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to PARAMETER LIST LENGTH ERROR. A PARAMETER LIST LENGTH set to zero specifies that no data shall be sent.

The CONTROL byte is defined in SAM-5.

### 5.28.2 UNMAP parameter list

The UNMAP parameter list (see table 97) contains the data sent by an application client along with an UNMAP command. Included in the data are an UNMAP parameter list header and block descriptors for LBA extents for which unmap requests (see 4.7.3.4.1) are to be processed by the device server. The LBAs specified in the block descriptors may contain overlapping extents, and may be in any order.

If the ANCHOR bit in the CDB is set to zero, and the logical unit is thin provisioned (see 4.7.3.3), then the logical block provisioning state for each specified LBA:

a) should become deallocated;
b) may become anchored; or
c) may remain unchanged.

If:

a) the ANCHOR bit in the CDB is set to one, and the ANC_SUP field in the Logical Block Provisioning VPD page (see 6.6.4) is set to one; or
b) the logical unit is resource provisioned,

then, for each specified LBA:

a) if the LBA is mapped, then the LBA shall not become deallocated, and:
   A) that LBA should become anchored (see 4.7.4.3.3); or
   B) the logical block provisioning state of that LBA may remain unchanged;
b) if the LBA is deallocated, then that LBA shall become anchored (see 4.7.4.4.3). If a lack of LBA mapping resources prevents the LBA from becoming anchored, then the device server shall terminate the command as described in 4.7.3.7; or
c) if the LBA is anchored, then that LBA shall remain anchored.

**Table 97 — Unmap parameter list**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | UNMAP DATA LENGTH (n - 1) | | | | |
| 1 | | | | | | | | (LSB) |
| 2 | (MSB) | | | UNMAP BLOCK DESCRIPTOR DATA LENGTH (n - 7) | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | | | | Reserved | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| **UNMAP block descriptor list** | | | | | | | | |
| 8 | | | | UNMAP block descriptor [first] (see table 98) | | | | |
| ••• | | | | | | | | |
| 23 | | | | | | | | |
| | | | | ⋮ | | | | |
| n - 15 | | | | UNMAP block descriptor [last] (see table 98) | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | |

The UNMAP DATA LENGTH field specifies the length in bytes of the following data that is available to be transferred from the Data-Out Buffer. The unmap data length does not include the number of bytes in the UNMAP DATA LENGTH field.

The UNMAP BLOCK DESCRIPTOR DATA LENGTH field specifies the length in bytes of the UNMAP block descriptors that are available to be transferred from the Data-Out Buffer. The unmap block descriptor data length should be a multiple of 16. If the unmap block descriptor data length is not a multiple of 16, then the last unmap block descriptor is incomplete and shall be ignored. If the UNMAP BLOCK DESCRIPTOR DATA LENGTH is set to zero, then no unmap block descriptors are included in the UNMAP parameter list. This condition shall not be considered an error.

If any UNMAP block descriptors in the UNMAP block descriptor list are truncated due to the parameter list length in the CDB, then those UNMAP block descriptors shall be ignored.

Table 98 defines an UNMAP block descriptor.

**Table 98 — UNMAP block descriptor**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | | | | UNMAP LOGICAL BLOCK ADDRESS | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | (MSB) | | | | | | | |
| ••• | | | | NUMBER OF LOGICAL BLOCKS | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | | | | | | | | |
| ••• | | | | Reserved | | | | |
| 15 | | | | | | | | |

The UNMAP LOGICAL BLOCK ADDRESS field specifies the first LBA of the UNMAP block descriptor that is requested to be unmapped (see 4.7.3.4.1).

The NUMBER OF LOGICAL BLOCKS field specifies the number of LBAs that are requested to be unmapped beginning with the LBA specified by the UNMAP LOGICAL BLOCK ADDRESS field.

If the NUMBER OF LOGICAL BLOCKS is set to zero, then no LBAs shall be unmapped for this UNMAP block descriptor. This condition shall not be considered an error.

If the specified LBA and the specified number of logical blocks exceeds the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

If the total number of logical blocks specified in the UNMAP block descriptor data exceeds the value indicated in the MAXIMUM UNMAP LBA COUNT field in the Block Limits VPD page (see 6.6.3), or if the number of UNMAP block descriptors exceeds the value of the MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT field in the Block Limits VPD page, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

### 5.29 VERIFY (10) command

The VERIFY (10) command (see table 100) requests that the device server:

a) perform write medium operations on any logical blocks in cache that are referenced by the LBA(s) specified by this command (i.e., as would be done in response to a SYNCHRONIZE CACHE

command (see 5.26) with:
A) the SYNC_NV bit set to zero;
B) the LOGICAL BLOCK ADDRESS field set to the value of the VERIFY command's LOGICAL BLOCK ADDRESS field; and
C) the NUMBER OF LOGICAL BLOCKS field set to the value of the VERIFY command's VERIFICATION LENGTH field);

b) perform verify medium operations on the logical blocks that are referenced by the LBA(s) specified by this command; and
c) if specified, perform a compare operation on:
A) the logical block data transferred from the Data-Out Buffer; and
B) the logical block data from the verify medium operations.

Table 99 defines the Data-Out Buffer contents for the VERIFY (10) command based on the setting of the BYTCHK field in the CDB.

**Table 99 — Data-Out Buffer contents for the VERIFY (10) command**

| BYTCHK field | Data-Out buffer contents |
|---|---|
| 00b | Not used |
| 01b | Logical block data for the number of logical blocks specified in the VERIFICATION LENGTH field |
| 10b | Not defined |
| 11b | Logical block data for a single logical block |

NOTE 23 - Migration from the VERIFY (10) command to the VERIFY (16) command is recommended for all implementations.

**Table 100 — VERIFY (10) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 100 for the VERIFY (10) command.

See the READ (10) command (see 5.11) for the definition of the DPO bit. See the PRE-FETCH (10) command (see 5.8) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.8) and 4.23 for the definition of the GROUP NUMBER field.

If the byte check (BYTCHK) field is set to 00b, then:

a) for any mapped LBA specified by the command, after performing the verify medium operation, the device server shall check the protection information from the verify medium operation based on the

VRPROTECT field as defined in table 101; and

b) for any unmapped LBA specified by the command, a verify medium operation shall complete without error.

If:

a) the BYTCHK field is set to 01b or 11b;
b) the VBULS bit is set to zero in the Block Device Characteristics VPD page (see 6.6.2); and
c) any LBA specified by the command is unmapped (i.e., deallocated or anchored),

then the device server shall terminate the command with CHECK CONDITION status with the sense key set to MISCOMPARE and the additional sense code set to MISCOMPARE VERIFY OF UNMAPPED LBA.

If:

a) the BYTCHK field is set to 01b or 11b; and
b) either:
   A) the VBULS bit is set to one in the Block Device Characteristics VPD page; or
   B) all LBAs specified by the command are mapped,

then:

a) the compare operation of the logical block data shall include:
   A) a compare operation of:
      a) user data from the verify medium operation; and
      b) user data transferred from the Data-Out Buffer;

      and
   B) a compare operation based on the VRPROTECT field as defined in table 104 of:
      a) protection information from the verify medium operation; and
      b) protection information transferred from the Data-Out Buffer;

   and
b) for any mapped LBA specified by the command, the compare operation of the logical block data shall be accompanied by:
   A) a check of protection information from the verify medium operation based on the VRPROTECT field as defined in table 102; and
   B) a check of protection information transferred from the Data-Out Buffer based on the VRPROTECT field as defined in table 103.

The order of the user data and protection information checks and compare operations is vendor specific.

If a compare operation indicates a miscompare, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to MISCOMPARE and the additional sense code set to the appropriate value for the condition.

The VERIFICATION LENGTH field specifies the number of contiguous logical blocks that shall be verified, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. A VERIFICATION LENGTH field set to zero specifies that no logical blocks shall be transferred or verified. This condition shall not be considered an error. If the specified LBA and the specified verification length exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The VERIFICATION LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.6.3).

If the BYTCHK field is set to 01b, then the VERIFICATION LENGTH field also specifies the number of logical blocks that the device server shall transfer from the Data-Out Buffer.

The CONTROL byte is defined in SAM-5.

If the BYTCHK field is set to 00b, then table 101 defines the checks that the device server shall perform on the protection information from the verify medium operation based on the VRPROTECT field. All footnotes for

table 101 are at the end of the table.

**Table 101 — VRPROTECT field with the BYTCHK field set to 00b – checking protection information from a verify medium operation** (part 1 of 3)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d] [e], additional sense code |
|---|---|---|---|---|
| 000b | Yes [i] | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | GRD_CHK = 0 | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | No protection information on the medium to check. Only user data is checked. | | |
| 001b<br>101b [b] | Yes | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | GRD_CHK = 0 | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | Error condition [a] | | |
| 010b [b] | Yes | LOGICAL BLOCK GUARD | No check performed | |
| | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | Error condition [a] | | |

**Table 101 — VRPROTECT field with the BYTCHK field set to 00b – checking protection information from a verify medium operation** (part 2 of 3)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d] [e], additional sense code |
|---|---|---|---|---|
| 011b [b] | Yes | LOGICAL BLOCK GUARD | No check performed | |
| | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No | Error condition [a] | | |
| 100b [b] | Yes | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | GRD_CHK = 0 | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No | Error condition [a] | | |
| 110b to 111b | Reserved | | | |

**Table 101 — VRPROTECT field with the BYTCHK field set to 00b – checking protection information from a verify medium operation** (part 3 of 3)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d] [e], additional sense code |
|---|---|---|---|---|

[a] If the logical unit supports protection information (see 4.22) but has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

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

[c] If the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field, then the device server shall check each logical block application tag. If the ATO bit in the Control mode page (see SPC-4) is set to one, then this knowledge is acquired from:

a) the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB, if the command is a VERIFY (32) command (see 5.32);
b) the Application Tag mode page (see 6.5.3), if the command is not a VERIFY (32) command and the ATMPE bit in the Control mode page (see SPC-4) is set to one; or
c) a method not defined by this standard, if the command is not a VERIFY (32) command and the ATMPE bit is set to zero.

[d] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

[e] If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

[f] See the Extended INQUIRY Data VPD page (see SPC-4) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bit.

[g] If the device server detects:

a) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.22.2.3) or type 2 protection (see 4.22.2.4) is enabled; or
b) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.22.2.5) is enabled,

then the device server shall not check any protection information in the associated protection information interval.

[h] If type 1 protection is enabled, then the device server shall check each logical block reference tag by comparing it to the lower four bytes of the LBA associated with the logical block. If type 2 protection or type 3 protection is enabled, and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server shall check the logical block reference tag. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 5.32). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

[i] If the DPICZ bit in the Control mode page (see SPC-4) is set to one, then protection information shall not be checked.

If the BYTCHK field is set to 01b or 11b, then table 102 defines the checks that the device server shall perform on the protection information from the verify medium operation based on the VRPROTECT field. All footnotes for

table 102 are at the end of the table.

**Table 102 — VRPROTECT field with the BYTCHK field set to 01b or 11b – checking protection information from the verify medium operation** (part 1 of 2)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d e], additional sense code |
|---|---|---|---|---|
| 000b | Yes [i] | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | GRD_CHK = 0 | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c g] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | No protection information available to check | | |
| 001b<br>010b<br>011b<br>100b<br>101b [b] | Yes | LOGICAL BLOCK GUARD | No check performed | |
| | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No | Error condition [a] | | |
| 110b<br>to<br>111b | Reserved | | | |

**Table 102 — VRPROTECT field with the BYTCHK field set to 01b or 11b – checking protection information from the verify medium operation** (part 2 of 2)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d] [e], additional sense code |
|---|---|---|---|---|

[a] If the logical unit supports protection information (see 4.22) but has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

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

[c] If the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field, then the device server shall check each logical block application tag. If the ATO bit in the Control mode page (see SPC-4) is set to one, then this knowledge is acquired from:

a) the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB, if the command is a VERIFY (32) command (see 5.32);
b) the Application Tag mode page (see 6.5.3), if the command is not a VERIFY (32) command and the ATMPE bit in the Control mode page (see SPC-4) is set to one; or
c) a method not defined by this standard, if the command is not a VERIFY (32) command and the ATMPE bit is set to zero.

[d] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

[e] If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

[f] See the Extended INQUIRY Data VPD page (see SPC-4) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bit.

[g] If the device server detects:

a) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.22.2.3) or type 2 protection (see 4.22.2.4) is enabled; or
b) a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 4.22.2.5) is enabled,

then the device server shall not check any protection information in the associated protection information interval.

[h] If type 1 protection is enabled, then the device server shall check each logical block reference tag by comparing it to the lower four bytes of the LBA associated with the logical block. If type 2 protection or type 3 protection is enabled, and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server shall check the logical block reference tag. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 5.32). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

[i] If the DPICZ bit in the Control mode page (see SPC-4) is set to one, then protection information shall not be checked.

If the BYTCHK field is set to 01b or 11b, then table 103 defines the checks that the device server shall perform on the protection information transferred from the Data-Out Buffer based on the VRPROTECT field. All footnotes

for table 103 are at the end of the table.

**Table 103 — VRPROTECT field with the BYTCHK field set to 01b or 11b – checking protection information from the Data-Out Buffer** (part 1 of 2)

| Code | Logical unit formatted with protection information | Field in protection information | Device server check | If check fails [d] [e], additional sense code |
|---|---|---|---|---|
| 000b | Yes | No protection information in the Data-Out Buffer to check | | |
| | No | No protection information in the Data-Out Buffer to check | | |
| 001b [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | May [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | Shall (except for type 3) [f] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [a] | | |
| 010b [b] | Yes | LOGICAL BLOCK GUARD | Shall not | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | May [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | May [f] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [a] | | |
| 011b [b] | Yes | LOGICAL BLOCK GUARD | Shall not | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | Shall not | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | Shall not | No check performed |
| | No | Error condition [a] | | |
| 100b [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | Shall not | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | Shall not | No check performed |
| | No | Error condition [a] | | |

**Table 103 — VRPROTECT field with the BYTCHK field set to 01b or 11b – checking protection information from the Data-Out Buffer** (part 2 of 2)

| Code | Logical unit formatted with protection information | Field in protection information | Device server check | If check fails [d] [e], additional sense code |
|---|---|---|---|---|
| 101b [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | May [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | May [f] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [a] | | |
| 110b to 111b | Reserved | | | |

[a] If the logical unit supports protection information (see 4.22) but has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

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

[c] If the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field, and the ATO bit is set to one in the Control mode page (see SPC-4), then the device server may check each logical block application tag. If the ATO bit is set to one, then this knowledge is acquired from:

a) the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB, if the command is a VERIFY (32) command;
b) the Application Tag mode page (see 6.5.3), if the command is not a VERIFY (32) command and the ATMPE bit in the Control mode page (see SPC-4) is set to one; or
c) a method not defined by this standard, if the command is not a VERIFY (32) command and the ATMPE bit is set to zero.

[d] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

[e] If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

[f] If type 1 protection is enabled, then the device server shall check the logical block reference tag by comparing it to the lower four bytes of the LBA associated with the logical block. If type 2 protection is enabled, and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server shall check each logical block reference tag. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 5.32). If type 3 protection is enabled, the ATO bit is set to one in the Control mode page (see SPC-4), and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server may check each logical block reference tag. If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

If the BYTCHK field is set to 01b or 11b, then table 104 defines the processing by the device server of the protection information during the compare operation based on the VRPROTECT field. All footnotes for table 104 are at the end of the table.

**Table 104 — VRPROTECT field with the BYTCHK field set to 01b or 11b – compare operation requirements**
(part 1 of 3)

| Code | Logical unit formatted with protection information | Field in protection information | Compare operation | If compare fails [c] [d], additional sense code |
|---|---|---|---|---|
| 000b | Yes | No protection information in the Data-Out Buffer to compare. Only user data is compared within each logical block. | | |
| | No | No protection information from the verify medium operation or in the Data-Out Buffer to compare. Only user data is compared within each logical block. | | |
| 001b [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 1) [e] | Shall | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f] | Shall not | No compare performed |
| | | LOGICAL BLOCK REFERENCE TAG (not type 3) | Shall [g] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 1) | Shall [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 0) | Shall not | No compare performed |
| | No | Error condition [a] | | |

**Table 104 — VRPROTECT field with the BYTCHK field set to 01b or 11b – compare operation requirements**
(part 2 of 3)

| Code | Logical unit formatted with protection information | Field in protection information | Compare operation | If compare fails [c] [d], additional sense code |
|---|---|---|---|---|
| 010b [b] | Yes | LOGICAL BLOCK GUARD | Shall not | No compare performed |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 1) [e] | Shall | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f] | Shall not | No compare performed |
| | | LOGICAL BLOCK REFERENCE TAG (not type 3) | Shall [g] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 1) | Shall [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 0) | Shall not | No compare performed |
| | No | Error condition [a] | | |
| 011b 100b [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 1) [e] | Shall | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f] | Shall not | No compare performed |
| | | LOGICAL BLOCK REFERENCE TAG (not type 3) | Shall [g] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 1) | Shall [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 0) | Shall not | No compare performed |
| | No | Error condition [a] | | |

**Table 104 — VRPROTECT field with the BYTCHK field set to 01b or 11b – compare operation requirements**
(part 3 of 3)

| Code | Logical unit formatted with protection information | Field in protection information | Compare operation | If compare fails [c] [d], additional sense code |
|---|---|---|---|---|
| 101b [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 1) [e] | Shall | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f] | Shall not | No compare performed |
| | | LOGICAL BLOCK REFERENCE TAG | Shall not | No compare performed |
| | No | Error condition [a] | | |
| 110b to 111b | Reserved | | | |

[a] If the logical unit supports protection information (see 4.22) but has not been formatted with protection information, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

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

[c] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to MISCOMPARE.

[d] If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

[e] If the ATO bit is set to one in the Control mode page (see SPC-4), then the device server shall not modify the logical block application tag.

[f] If the ATO bit is set to zero in the Control mode page (see SPC-4), then the device server may modify any logical block application tag.

[g] If the BYTCHK field is set to 11b, then the device server shall compare the value from each LOGICAL BLOCK REFERENCE TAG field received in the single logical block data from the Data-Out Buffer with the corresponding LOGICAL BLOCK REFERENCE TAG field in the first logical block read from the medium, and the device server shall compare the value of the previous LOGICAL BLOCK REFERENCE TAG field plus one with each of the subsequent LOGICAL BLOCK REFERENCE TAG fields. (see 4.22.3).

[h] If the BYTCHK field is set to 11b, then the device server shall compare the value from each LOGICAL BLOCK REFERENCE TAG field received in the single logical block data from the Data-Out Buffer with the corresponding LOGICAL BLOCK REFERENCE TAG field in each logical block read from the medium (see 4.22.3),

### 5.30 VERIFY (12) command

The VERIFY (12) command (see table 105) requests that the device server perform the actions defined for the VERIFY (10) command (see 5.29).

NOTE 24 - Migration from the VERIFY (12) command to the VERIFY (16) command is recommended for all implementations.

**Table 105 — VERIFY (12) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 105 for the VERIFY (12) command.

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

### 5.31 VERIFY (16) command

The VERIFY (16) command (see table 106) requests that the device server perform the actions defined for the VERIFY (10) command (see 5.29).

**Table 106 — VERIFY (16) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 106 for the VERIFY (16) command.

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

### 5.32 VERIFY (32) command

The VERIFY (32) command (see table 107) requests that the device server perform the actions defined for the VERIFY (10) command (see 5.29).

The device server shall process a VERIFY (32) command only if type 2 protection is enabled (see 4.22.2.4).

**Table 107 — VERIFY (32) command**

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

The OPERATION CODE byte, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-4 and shall be set to the values shown in table 107 for the VERIFY (32) command.

See the VERIFY (10) command (see 5.29) for the definitions of the CONTROL byte, the GROUP NUMBER field, the VRPROTECT field, the DPO bit, the BYTCHK field, the LOGICAL BLOCK ADDRESS field, and the VERIFICATION LENGTH field.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 101, table 102, table 103, and table 104 in 5.29), the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.22.3).

If the ATO bit is set to one in the Control mode page (see SPC-4), and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 101, table 102, table 103, and table 104 in 5.29), then the LOGICAL

BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in every instance of the protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of the protection information.

The LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored if:

a) the ATO bit is set to zero; or
b) the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 101, table 102, table 103, and table 104 in 5.29).

### 5.33 WRITE (10) command

The WRITE (10) command (see table 108) requests that the device:

a) transfer the logical block data specified by this command from the Data-Out Buffer;
b) write the transferred logical blocks to the LBAs specified by this command based on the rules for caching (see 4.15).

NOTE 25 - Migration from the WRITE (10) command to the WRITE (16) command is recommended for all implementations.

**Table 108 — WRITE (10) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 108 for the WRITE (10) command.

The device server shall check the protection information, if any, transferred from the Data-Out Buffer based on the WRPROTECT field as described in table 109. All footnotes for table 109 are at the end of the table.

**Table 109 — WRPROTECT field** (part 1 of 3)

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

**Table 109 — WRPROTECT field** (part 2 of 3)

| Code | Logical unit formatted with protection information | Field in protection information | Device server check | If check fails [d] [i], additional sense code |
|---|---|---|---|---|
| 001b [b] | Yes [e] | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | Dependent on RWWP [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | Shall (except for type 3) [j] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No [a] | No protection information available to check | | |
| 010b [b] | Yes [e] | LOGICAL BLOCK GUARD | Shall not | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | Dependent on RWWP [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | May [j] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No [a] | No protection information available to check | | |
| 011b [b] | Yes [e] | LOGICAL BLOCK GUARD | Shall not | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | Shall not | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | Shall not | No check performed |
| | No [a] | No protection information available to check | | |
| 100b [b] | Yes [e] | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | Shall not | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | Shall not | No check performed |
| | No [a] | No protection information available to check | | |
| 101b [b] | Yes [e] | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | Dependent on RWWP [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | May [j] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No [a] | No protection information available to check | | |
| 110b to 111b | Reserved | | | |

**Table 109 — WRPROTECT field** (part 3 of 3)

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

[a] If a logical unit supports protection information (see 4.22) but has not been formatted with protection information, then the device server shall terminate a command specifying a write medium operation to the logical unit with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

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

[c] If the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field, and the ATO bit is set to one in the Control mode page (see SPC-4), then the device server:

a) may check each logical block application tag if the RWWP bit is set to zero in the Control mode page (see SPC-4); and

b) shall check each logical block application tag if the RWWP bit is set to one in the Control mode page.

If the ATO bit in the Control mode page (see SPC-4) is set to one, then this knowledge is acquired from:

a) the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB, if a WRITE (32) command (see 5.36) is received by the device server;

b) the Application Tag mode page (see 6.5.3), if a command other than WRITE (32) is received by the device server and the ATMPE bit in the Control mode page (see SPC-4) is set to one or

c) a method not defined by this standard, if a command other than WRITE (32) is received by the device server, and the ATMPE bit is set to zero.

[d] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

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

[f] The device server shall write a generated CRC (see 4.22.4.2) into each LOGICAL BLOCK GUARD field.

[g] If the RWWP bit in the Control mode page (see SPC-4) is set to one, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. If the RWWP bit is set to zero, and:

a) type 1 protection is enabled, then the device server shall write the least significant four bytes of each LBA into the LOGICAL BLOCK REFERENCE TAG field of each of the written logical blocks; or

b) type 2 protection or type 3 protection is enabled, then the device server shall write a value of FFFF_FFFFh into the LOGICAL BLOCK REFERENCE TAG field of each of the written logical blocks.

[h] If the ATO bit is set to one in the Control mode page (see SPC-4), then the device server shall write FFFFh into each LOGICAL BLOCK APPLICATION TAG field. If the ATO bit is set to zero, then the device server may write any value into each LOGICAL BLOCK APPLICATION TAG field.

[i] If multiple errors occur while the device server is processing the command, then the selection by the device server of which error to report is not defined by this standard.

[j] If type 1 protection is enabled, then the device server shall check the logical block reference tag by comparing it to the lower four bytes of the LBA associated with the logical block. If type 2 protection is enabled, and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server shall check each logical block reference tag. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a WRITE (32) command (see 5.36). If type 3 protection is enabled, the ATO bit is set to one in the Control mode page (see SPC-4), and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server may check each logical block reference tag. If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

See the READ (10) command (see 5.11) for the definition of the DPO bit.

The force unit access (FUA) bit and the force unit access non-volatile cache (FUA_NV) bit are defined in table 110.

**Table 110 — Force unit access for writes**

| FUA | FUA_NV | Description |
|---|---|---|
| 0 | 0 | The device server shall write the logical blocks to volatile cache, non-volatile cache, and/or the medium. |
| 0 | 1 | If the NV_SUP bit is set to one in the Extended INQUIRY Data VPD page (see SPC-4), then the device server shall write the logical blocks to non-volatile cache and/or the medium.<br><br>If the NV_SUP bit is set to zero in the Extended INQUIRY Data VPD page (see SPC-4), then the device server shall write the logical blocks to volatile cache, non-volatile cache, and/or the medium. |
| 1 | 0 or 1 | The device server shall write the logical blocks to the medium, and shall not complete the command with GOOD status until the logical blocks have been written on the medium without error. |

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

See the PRE-FETCH (10) command (see 5.8) for the definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command and 4.23 for the definition of the GROUP NUMBER field.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred from the Data-Out Buffer and written, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. A TRANSFER LENGTH field set to zero specifies that no logical blocks shall be transferred or written. This condition shall not be considered an error. Any other value specifies the number of logical blocks that shall be transferred and written. If the specified LBA and the specified transfer length exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.6.3).

The CONTROL byte is defined in SAM-5.

### 5.34 WRITE (12) command

The WRITE (12) command (see table 111) requests that the device server perform the actions defined for the WRITE (10) command (see 5.33).

NOTE 26 - Migration from the WRITE (12) command to the WRITE (16) command is recommended for all implementations.

**Table 111 — WRITE (12) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 111 for the WRITE (12) command.

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

### 5.35 WRITE (16) command

The WRITE (16) command (see table 112) requests that the device server perform the actions defined for the WRITE (10) command (see 5.33).

**Table 112 — WRITE (16) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 112 for the WRITE (16) command.

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

### 5.36 WRITE (32) command

The WRITE (32) command (see table 113) requests that the device server perform the actions defined for the WRITE (10) command (see 5.33).

The device server shall process a WRITE (32) command only if type 2 protection is enabled (see 4.22.2.4).

**Table 113 — WRITE (32) command**

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

The OPERATION CODE byte, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-4 and shall be set to the values shown in table 113 for the WRITE (32) command.

See the WRITE (10) command (see 5.33) for the definitions of the CONTROL byte, the GROUP NUMBER field, the WRPROTECT field, the DPO bit, the FUA bit, the FUA_NV bit, the LOGICAL BLOCK ADDRESS field, and the TRANSFER LENGTH field.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 109 in 5.33), the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.22.3).

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 109 in 5.33), then the LOGICAL BLOCK APPLICATION TAG MASK field

contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information.

The LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored if:

a) the ATO bit is set to zero; or
b) the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 109 in 5.33).

### 5.37 WRITE AND VERIFY (10) command

The WRITE AND VERIFY (10) command (see table 114) requests that the device server:

1) transfer all of the logical block data for the command from the Data-Out Buffer (i.e., logical block data referenced by any LBAs specified for the command that is already in cache shall not be used and shall become invalid);
2) perform write medium operations on the logical blocks referenced by the LBA(s) specified by this command;
3) perform verify medium operations on the logical blocks that are referenced by the LBA(s) specified by this command; and
4) if specified, perform a compare operation on:
   A) the logical block data transferred from the Data-Out Buffer; and
   B) the logical block data from the verify medium operations.

The device server may process the LBAs in any order but shall perform this sequence in the specified order for a given LBA.

NOTE 27 - Migration from the WRITE AND VERIFY (10) command to the WRITE AND VERIFY (16) command is recommended for all implementations.

**Table 114 — WRITE AND VERIFY (10) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 114 for the WRITE AND VERIFY (10) command.

See the PRE-FETCH (10) command (see 5.8) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command and 4.23 for the definition of the GROUP NUMBER field. See the WRITE (10) command (see 5.33) for the definitions of the CONTROL byte, the TRANSFER LENGTH field and the WRPROTECT field. See the READ (10) command (see 5.11) for the definition of the DPO bit.

See the VERIFY (10) command (see 5.29) for definition of the byte check (BYTCHK) field when set to 00b, 01b, and 10b. For a WRITE AND VERIFY (10) command, a BYTCHK field set to 11b is reserved.

### 5.38 WRITE AND VERIFY (12) command

The WRITE AND VERIFY (12) command (see table 115) requests that the device server perform the actions defined for the WRITE AND VERIFY (10) command (see 5.37).

NOTE 28 - Migration from the WRITE AND VERIFY (12) command to the WRITE AND VERIFY (16) command is recommended for all implementations.

**Table 115 — WRITE AND VERIFY (12) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 115 for the WRITE AND VERIFY (12) command.

See the WRITE AND VERIFY (10) command (see 5.37) for the definitions of the other fields in this command.

### 5.39 WRITE AND VERIFY (16) command

The WRITE AND VERIFY (16) command (see table 116) requests that the device server perform the actions defined for the WRITE AND VERIFY (10) command (see 5.37).

**Table 116 — WRITE AND VERIFY (16) command**

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

The OPERATION CODE byte is defined in SPC-4 and is set to the value shown in table 116 for the WRITE AND VERIFY (16) command.

See the WRITE AND VERIFY (10) command (see 5.37) for the definitions of the other fields in this command.

### 5.40 WRITE AND VERIFY (32) command

The WRITE AND VERIFY (32) command (see table 117) requests that the device server perform the actions defined for the WRITE AND VERIFY (10) command (see 5.37).

**Table 117 — WRITE AND VERIFY (32) command**

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

The OPERATION CODE byte, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-4 and shall be set to the values shown in table 117 for the WRITE AND VERIFY (32) command.

See the WRITE AND VERIFY (10) command (see 5.37) for the definitions of the CONTROL byte, the GROUP NUMBER field, the WRPROTECT field, the DPO bit, the BYTCHK field, the LOGICAL BLOCK ADDRESS field, and the TRANSFER LENGTH field.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 109 in 5.33), the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.22.3).

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 109 in 5.33), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in

every instance of protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information.

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 109 in 5.33), or if the ATO bit is set to zero, then the LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored.

### 5.41 WRITE LONG (10) command

The WRITE LONG (10) command (see table 118) requests that the device server mark a logical block or physical block as containing a pseudo unrecovered error, or transfer data for a single logical block or physical block from the Data-Out Buffer and write it to the medium in the logical unit. The data written shall be the same length and shall be in the same order as the data returned by the READ LONG (10) command (see 5.19). The device server shall write the logical block or physical block to the medium, and shall not complete the command with GOOD status until the logical block has been written on the medium without error.

NOTE 29 - Migration from the WRITE LONG (10) command to the WRITE LONG (16) command is recommended for all implementations.

**Table 118 — WRITE LONG (10) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 118 for the WRITE LONG (10) command.

The correction disabled (COR_DIS) bit, the write uncorrectable error (WR_UNCOR) bit, and the physical block (PBLOCK) bit are defined in table 119. If there is more than one logical block per physical block (i.e., the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field in the READ CAPACITY (16) parameter data (see 5.16.1) is set to a non-zero value), then the device server shall support the WR_UNCOR bit and the PBLOCK bit.

**Table 119 — COR_DIS bit, WR_UNCOR bit, and PBLOCK bit** (part 1 of 2)

| COR_DIS | WR_UNCOR | PBLOCK | More than one logical block per physical block [a] | Description |
|---|---|---|---|---|
| 0 | 0 | 0 | yes or no | Write only the specified logical block using the value in the BYTE TRANSFER LENGTH field. |
| | | 1 | no | Terminate the WRITE LONG command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. |
| | | | yes | Write the entire physical block containing the specified logical block using the value in the BYTE TRANSFER LENGTH field. |
| 0 | 1 | 0 | yes or no | Mark only the specified logical block as containing a pseudo unrecovered error with correction enabled (see 4.18.2) in a manner that causes the device server to perform the maximum error recovery as defined by the Read-Write Error Recovery mode page (see 6.5.8).<br><br>Ignore the BYTE TRANSFER LENGTH field, and transfer no data. |
| | | 1 | no | Terminate the WRITE LONG command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. |
| | | | yes | Mark the entire physical block containing the specified logical block as containing a pseudo unrecovered error with correction enabled (i.e., mark all of the logical blocks in the same physical block that contains the specified logical block as containing a pseudo unrecovered error with correction enabled) (see 4.18.2).<br><br>Ignore the BYTE TRANSFER LENGTH field, and transfer no data. |

[a] An entry of “yes” means that the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field in the READ CAPACITY (16) parameter data (see 5.16.2) is set to a non-zero value. An entry of “no” means that the field is set to zero.

**Table 119 — COR_DIS bit, WR_UNCOR bit, and PBLOCK bit** (part 2 of 2)

| COR_DIS | WR_UNCOR | PBLOCK | More than one logical block per physical block [a] | Description |
|---|---|---|---|---|
| 1 | 0 | 0 | yes or no | Mark only the specified logical block as containing a pseudo unrecovered error with correction disabled (see 4.18.2).<br><br>Write only the specified logical block using the value in the BYTE TRANSFER LENGTH field. |
| | | 1 | no | Terminate the WRITE LONG command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. |
| | | | yes | Mark the entire physical block containing the specified logical block as containing a pseudo unrecovered error with correction disabled (i.e., mark all of the logical blocks in the same physical block that contains the specified logical block as containing a pseudo unrecovered error with correction disabled) (see 4.18.2).<br><br>Write the entire physical block containing the specified logical block using the value in the BYTE TRANSFER LENGTH field. |
| 1 | 1 | 0 | yes or no | Mark only the specified logical block as containing a pseudo unrecovered error with correction disabled (see 4.18.2).<br><br>Ignore the BYTE TRANSFER LENGTH field, and transfer no data. |
| | | 1 | no | Terminate the WRITE LONG command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. |
| | | | yes | Mark the entire physical block containing the specified logical block as containing a pseudo unrecovered error with correction disabled (i.e., mark all of the logical blocks in the same physical block that contains the specified logical block as containing a pseudo unrecovered error with correction disabled) (see 4.18.2).<br><br>Ignore the BYTE TRANSFER LENGTH field, and transfer no data. |

[a] An entry of “yes” means that the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field in the READ CAPACITY (16) parameter data (see 5.16.2) is set to a non-zero value. An entry of “no” means that the field is set to zero.

In the Extended INQUIRY Data VPD page (see SPC-4), the setting of the CD_SUP bit indicates whether or not the logical unit supports the COR_DIS bit being set to one, and the setting of the WU_SUP bit indicates whether or not the logical unit supports the WR_UNCOR bit being set to one.

The LOGICAL BLOCK ADDRESS field specifies an LBA. If the specified LBA exceeds the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

If table 119 defines that the value in the BYTE TRANSFER LENGTH field is used, then the BYTE TRANSFER LENGTH field specifies the number of bytes of data that the device server shall transfer from the Data-Out Buffer and write to the specified logical block or physical block. If the BYTE TRANSFER LENGTH field is not set to zero and does not match the data length that the device server returns for a READ LONG command, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. In the sense data (see 4.18 and SPC-4), the ILI BIT and the VALID bit shall be set to one, and the INFORMATION field shall be set to the difference (i.e., residue) of the requested length minus the actual length in bytes. Negative values shall be indicated by two's complement notation. If the BYTE TRANSFER LENGTH field is set to zero, then no bytes shall be written. This condition shall not be considered an error.

The CONTROL byte is defined in SAM-5.

### 5.42 WRITE LONG (16) command

The WRITE LONG (16) command (see table 120) requests that the device server mark a logical block or physical block as containing an error, or transfer data for a single logical block or physical block from the Data-Out Buffer and write it to the medium in the logical unit. The data written shall be the same length and shall be in the same order as the data returned by the READ LONG (16) command (see 5.20). The device server shall write the logical block or physical block to the medium, and shall not complete the command with GOOD status until the logical block has been written on the medium without error. This command is implemented as a service action of the SERVICE ACTION OUT operation code (see A.2).

**Table 120 — WRITE LONG (16) command**

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

The OPERATION CODE byte and SERVICE ACTION field are defined in SPC-4 and shall be set to the values shown in table 120 for the WRITE LONG (16) command.

See the WRITE LONG (10) command (see 5.41) for the definitions of the fields in this command.

### 5.43 WRITE SAME (10) command

The WRITE SAME (10) command (see table 121) requests that the device server transfer a single logical block from the Data-Out Buffer and write the contents of that logical block to the specified range of LBAs. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format.

The device server shall write the single block of user data received from the Data-Out Buffer to each logical block without modification.

If the medium is formatted with type 1 or type 2 protection information, then:

a) the device server shall place the value from each LOGICAL BLOCK REFERENCE TAG field received in the single block of data from the Data-Out Buffer into the corresponding LOGICAL BLOCK REFERENCE TAG field of the first logical block written to the medium. The device sever shall place the value of the previous LOGICAL BLOCK REFERENCE TAG field plus one into each of the subsequent LOGICAL BLOCK REFERENCE TAG fields;
b) if the ATO bit is set to one in the Control mode page (see SPC-4), then the device server shall place each logical block application tag received in the single block of data into the corresponding LOGICAL BLOCK APPLICATION TAG field of each logical block;
c) if the ATO bit is set to zero in the Control mode page, then the device server may write any value into the LOGICAL BLOCK APPLICATION TAG field(s) of each logical block; and
d) the device server shall place the value from each LOGICAL BLOCK GUARD field received in the single block of data from the Data-Out Buffer into the corresponding LOGICAL BLOCK GUARD field of each logical block.

If the medium is formatted with type 3 protection information, then:

a) if the ATO bit is set to one in the Control mode page (see SPC-4), then the device server shall place each logical block application tag received in the single block of data into the corresponding LOGICAL BLOCK REFERENCE TAG field of each logical block;
b) if the ATO bit is set to zero in the Control mode page, then the device server may write any value into the LOGICAL BLOCK REFERENCE TAG field(s) of each logical block; and
c) the device server shall place the value from each LOGICAL BLOCK GUARD field received in the single block of data from the Data-Out Buffer into the corresponding LOGICAL BLOCK GUARD field of each logical block.

NOTE 30 - Migration from the WRITE SAME (10) command to the WRITE SAME (16) command is recommended for all implementations.

**Table 121 — WRITE SAME (10) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 121 for the WRITE SAME (10) command.

See the WRITE (10) command (see 5.33) for the definition of the WRPROTECT field.

If the logical unit supports logical block provisioning management (see 4.7.3), then the ANCHOR bit, the UNMAP bit, and the ANC_SUP bit in the Logical Block Provisioning VPD page (see 6.6.4) determine how the device server processes the command as described in table 122.

**Table 122 — ANCHOR bit, UNMAP bit, and ANC_SUP bit relationships**

| UNMAP bit [a] | ANCHOR bit | ANC_SUP bit [b] | Action |
|---|---|---|---|
| 0b | 0b | 0 or 1 | Write [c] |
| | 1b | 0 or 1 | Error [d] |
| 1b | 0b | 0 or 1 | Unmap or write [e] |
| | 1b | 0 | Error [d] |
| | | 1 | Anchor or write [f] |

[a] The device server in a logical unit that supports logical block provisioning management (see 4.7.3) may implement the UNMAP bit.

[b] See the Logical Block Provisioning VPD page (see 6.6.4).

[c] The device server shall perform the specified write medium operation on each LBA specified by the command.

[d] The device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[e] If the Data-Out Buffer matches the logical block data returned by a read command for an unmapped LBA (see 4.7.4.2), then the device server in a thin provisioned logical unit should perform an unmap operation to deallocate each LBA specified by the command (see 4.7.4.4.1) but may perform an unmap operation to anchor each LBA specified by the command (see 4.7.4.5.1). If the Data-Out Buffer matches the logical block data returned by a read command for an unmapped LBA (see 4.7.4.2), then the device server in a resource provisioned logical unit should perform an unmap operation to anchor each LBA specified by the command. If the device server does not perform an unmap operation for an LBA, then the device server shall perform the specified write medium operation for that LBA (see 4.7.3.4.3) based on the rules for caching.

[f] If the Data-Out Buffer matches the logical block data returned by a read command for an unmapped LBA (see 4.7.4.2), then the device server should perform an unmap operation to anchor each LBA specified by the command (see 4.7.4.5.1). If the device server does not perform the unmap operation for an LBA, then the device server shall perform the specified write medium operation for that LBA (see 4.7.3.4.3) based on the rules for caching.

The device server shall ignore the UNMAP bit and the ANCHOR bit, or the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB if:

a) the logical unit is fully provisioned (i.e., the LBPME bit is set to zero in the READ CAPACITY (16) parameter data (see 5.16.2)); and
b) the UNMAP bit is set to one or the ANCHOR bit is set to one.

See the PRE-FETCH (10) command (see 5.8) for the definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command (see 5.8) and 4.23 for the definition of the GROUP NUMBER field.

The NUMBER OF LOGICAL BLOCKS field specifies the number of contiguous logical blocks that are requested be unmapped or written, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. If the WSNZ bit in the Block Limits VPD page (see 6.6.3) is set to zero, then a NUMBER OF LOGICAL BLOCKS field set to zero specifies that the device server shall write all of the logical blocks starting with the one referenced by the LBA specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium. If the WSNZ bit is set to one, and the NUMBER OF LOGICAL BLOCKS field is set to zero, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. If the specified LBA and the

specified number of logical blocks exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. If the MAXIMUM WRITE SAME LENGTH field in the Block Limits VPD page is non-zero, and the value in the NUMBER OF LOGICAL BLOCKS field exceeds the value indicated in the MAXIMUM WRITE SAME LENGTH field, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

See the WRITE (10) command (see 5.33) for the definition of the CONTROL byte.

### 5.44 WRITE SAME (16) command

The WRITE SAME (16) command (see table 123) requests that the device server transfer a single logical block from the Data-Out Buffer and write the contents of that logical block to the specified range of LBAs. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format.

See the WRITE SAME (10) command (see 5.43) for the definition of the function of this command.

**Table 123 — WRITE SAME (16) command**

| Bit<br>Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (93h) | | | | | | | |
| 1 | WRPROTECT | | | ANCHOR | UNMAP | Obsolete | Obsolete | Reserved |
| 2 | (MSB) | | | | | | | |
| ••• | LOGICAL BLOCK ADDRESS | | | | | | | |
| 9 | | | | | | | | (LSB) |
| 10 | (MSB) | | | | | | | |
| ••• | NUMBER OF LOGICAL BLOCKS | | | | | | | |
| 13 | | | | | | | | (LSB) |
| 14 | Reserved | | | GROUP NUMBER | | | | |
| 15 | CONTROL | | | | | | | |

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 123 for the WRITE SAME (16) command.

See the WRITE SAME (10) command (see 5.43) for the definitions of the other fields in this command.

### 5.45 WRITE SAME (32) command

The WRITE SAME (32) command (see table 124) requests that the device server transfer a single logical block from the Data-Out Buffer and write the contents of that logical block to the specified range of LBAs. Each logical block includes user data and may include protection information, based on the WRPROTECT field and the medium format.

See the WRITE SAME (10) command (see 5.43) for the definition of the function of this command.

**Table 124 — WRITE SAME (32) command**

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

The OPERATION CODE byte, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-4 and shall be set to the values shown in table 124 for the WRITE SAME (32) command.

See the WRITE SAME (10) command (see 5.43) for the definitions of the CONTROL byte, the GROUP NUMBER field, the WRPROTECT field, the LOGICAL BLOCK ADDRESS field, the NUMBER OF LOGICAL BLOCKS field, the UNMAP bit, and the ANCHOR bit.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 109 in 5.33), the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.22.3).

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 109 in 5.33), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in every instance of protection information.

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 109 in 5.33), or if the ATO bit is set to zero, then the LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored.

## 5.46 WRITE USING TOKEN command

### 5.46.1 WRITE USING TOKEN command overview

The WRITE USING TOKEN command (see table 125) requests that the copy manager (see SPC-4) write logical block data represented by the specified ROD token to the specified LBAs. Each logical block written includes user data, and may include protection information.

**Table 125 — WRITE USING TOKEN command**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (83h) | | | | | | | |
| 1 | Reserved | | | SERVICE ACTION (11h) | | | | |
| 2 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 5 | | | | | | | | |
| 6 | (MSB) | | | LIST IDENTIFIER | | | | |
| ••• | | | | | | | | |
| 9 | | | | | | | | (LSB) |
| 10 | (MSB) | | | PARAMETER LIST LENGTH | | | | |
| ••• | | | | | | | | |
| 13 | | | | | | | | (LSB) |
| 14 | Reserved | | | GROUP NUMBER | | | | |
| 15 | CONTROL | | | | | | | |

The OPERATION CODE byte and the SERVICE ACTION field are defined in SPC-4 and shall be set to the values shown in table 125 for the WRITE USING TOKEN command.

The LIST IDENTIFIER field is defined in SPC-4. The list identifier shall be processed as if the LIST ID USAGE field in the parameter data for an EXTENDED COPY(LID4) command (see SPC-4) is set to 00b.

The PARAMETER LIST LENGTH field specifies the length in bytes of the parameter data that is available to be transferred from the Data-Out Buffer. A PARAMETER LIST LENGTH set to zero specifies that no data shall be sent. This shall not be considered an error.

See the PRE-FETCH (10) command (see 5.8) and 4.23 for the definition of the GROUP NUMBER field.

The CONTROL byte is defined in SAM-5.

### 5.46.2 WRITE USING TOKEN parameter list

The parameter list for the WRITE USING TOKEN command is shown in table 126.

**Table 126 — WRITE USING TOKEN parameter data**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| 1 | WRITE USING TOKEN DATA LENGTH (n - 1) | | | | | | | (LSB) |
| 2 | Reserved | | | | | | DEL_TKN | IMMED |
| 3 | | | | | | | | |
| ••• | Reserved | | | | | | | |
| 7 | | | | | | | | |
| 8 | (MSB) | | | | | | | |
| ••• | OFFSET INTO ROD | | | | | | | |
| 15 | | | | | | | | (LSB) |
| 16 | (MSB) | | | | | | | |
| ••• | ROD TOKEN | | | | | | | |
| 527 | | | | | | | | (LSB) |
| 528 | | | | | | | | |
| ••• | Reserved | | | | | | | |
| 533 | | | | | | | | |
| 534 | BLOCK DEVICE RANGE DESCRIPTOR LENGTH (n - 535) | | | | | | | |
| 535 | | | | | | | | |
| **Block device range descriptor list** | | | | | | | | |
| 536 | Block device range descriptor [first] (see 5.7.3) | | | | | | | |
| ••• | | | | | | | | |
| 551 | | | | | | | | |
| | ⋮ | | | | | | | |
| n - 15 | Block device range descriptor [last] (see 5.7.3) | | | | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | |

The WRITE USING TOKEN DATA LENGTH field specifies the length in bytes of the data that is available to be transferred from the Data-Out Buffer. The write using token data length does not include the number of bytes in the WRITE USING TOKEN DATA LENGTH field. If the WRITE USING TOKEN DATA LENGTH field is less than 0226h (i.e., 550), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

An immediate (IMMED) bit set to zero specifies that the copy manager shall return status after the specified operations have completed. An IMMED bit set to one specifies that the copy manager shall return status after the entire parameter list has been transferred.

If the operations specified by a WRITE USING TOKEN command are processed as a background operation, then the copy manager shall not generate deferred errors (see SAM-5) to report the errors encountered, if any, during this processing. The copy manager shall make error information available to an application client using a RECEIVE ROD TOKEN INFORMATION command (see 5.22).

The delete token (DEL_TKN) bit specifies whether the ROD token specified in the ROD TOKEN field should be deleted when processing of the WRITE USING TOKEN command is complete. If the DEL_TKN bit is set to one, then the ROD token specified in the ROD TOKEN field should be deleted when processing of the WRITE USING TOKEN command is complete. If the DEL_TKN bit is set to zero, then the ROD token lifetime for the ROD token specified in the ROD TOKEN field shall be as described in SPC-4.

If the ROD TOKEN LENGTH field (see SPC-4) in the ROD TOKEN field is not set to 01F8h then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense key set to INVALID TOKEN OPERATION, INVALID TOKEN LENGTH.

The OFFSET INTO ROD field specifies the offset into the data represented by the ROD token from the first byte represented by the ROD token to the first byte to be transferred. The offset is specified in blocks based on the logical block size of the logical unit to which the WRITE USING TOKEN command is to write data. The copy manager that processes the WRITE USING TOKEN command shall compute the byte offset into the ROD by multiplying the contents of the OFFSET INTO ROD field by the logical block size of the logical unit to which the WRITE USING TOKEN command is to write data.

NOTE 31 - As an example of how to calculate the offset: a ROD token is created from logical blocks 15 to 20 followed by logical blocks 40 to 100 by a copy manager associated with a logical unit with a block size of 512 bytes per logical block. That ROD token is specified in a WRITE USING TOKEN command that transfers one logical block to a logical unit with a block size of 4 096 bytes per logical block. The subsequent RECEIVE ROD TOKEN INFORMATION command indicates the successful transfer of 4 096 bytes by setting the TRANSFER COUNT field to one. To create a WRITE USING TOKEN command that transfers bytes from the ROD token starting at the point where the previous WRITE USING TOKEN command stopped, the OFFSET INTO ROD field is set to one (i.e., the contents of the TRANSFER COUNT field) plus the value in the OFFSET INTO ROD field in the previous WRITE USING TOKEN command (i.e., zero). The copy manager multiplies one (i.e., the value in the OFFSET INTO ROD field) by 4 096 (i.e., the number of bytes in a logical block for the logical unit to which the data is being written) and the result is 4 096. As a result, the ROD token logical block that is the start of the transfer is logical block eight (i.e., LBA 42 from the logical unit whose block size is 512 bytes per logical block that was used to create the ROD token).

If the computed byte offset into the ROD is greater than or equal to the number of bytes represented by the ROD token, then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The ROD TOKEN field specifies the ROD token that represents the data from which logical block data is written to the media. The ROD token is defined as follows:

a) a ROD token returned by a RECEIVE ROD TOKEN INFORMATION command; or
b) a block device zero ROD token (see 4.30.4).

If the ROD token does not match any known to the copy manager, then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID TOKEN OPERATION, TOKEN UNKNOWN.

The BLOCK DEVICE RANGE DESCRIPTOR LENGTH field specifies the length in bytes of the block device range descriptor list. The block device range descriptor list length should be a multiple of 16. If the block device range descriptor list length is not a multiple of 16, then the last block device range descriptor is incomplete and shall be ignored. If the block device range descriptor list length is less than 16, then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the number of complete block device range descriptors is larger than the maximum range descriptors value in the Block Device ROD Token Limits descriptor (see 6.6.6.3), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to TOO MANY SEGMENT DESCRIPTORS.

If the same LBA is included in more than one block device range descriptor, then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

If the number of bytes of user data represented by the sum of the contents of the NUMBER OF LOGICAL BLOCKS fields in all of the complete block device range descriptors is larger than the maximum bytes in block ROD value in the Block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page (see SPC-4), then the copy manager shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. If the maximum bytes in block ROD value in the Block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page is not reported, then the maximum token transfer size value in the Block Device ROD Token Limits descriptor may indicate a different value for the maximum.

If the number of bytes of user data represented by the sum of the contents of the NUMBER OF LOGICAL BLOCKS fields in all of the complete block device range descriptors is larger than the number of bytes in the data represented by the ROD token minus the computed byte offset into the ROD (i.e., the total requested length of the transfer exceeds the length of the data available in the data represented by the ROD token), then the copy manager shall:

a) transfer as many whole logical blocks as possible; and
b) if any portion of a logical block that is written by the copy manager corresponds to offsets into the ROD at or beyond the length of the data represented by the ROD token, write that portion of the logical block with user data with all bits set to zero.

The copy manager may perform this check during the processing of each block device range descriptor.

### 5.47 XDWRITEREAD (10) command

The XDWRITEREAD (10) command (see table 127) requests that the device server perform the following as an uninterrupted sequence of actions (see 4.27):

1) if required by the rules for caching (see 4.15), perform read medium operations on the specified logical block(s) referenced by the LBA(s) specified by this command;
2) transfer the specified number of logical blocks from the Data-Out Buffer;
3) perform an XOR operation on:
   A) the user data contained in the logical blocks read from the medium or cached; and
   B) the user data contained in the logical blocks transferred from the Data-Out Buffer;
4) store the resulting XOR data in a buffer;
5) if the DISABLE WRITE bit is set to zero, then perform write medium operations on the logical blocks transferred from the Data-Out Buffer to the LBA(s) specified by this command and based on the rules for caching; and
6) transfer the resulting XOR data to the Data-In Buffer.

Logical block data for this command may include protection information, based on the WRPROTECT field, the XORPINFO bit, and the medium format. This command is only available on transport protocols supporting bidirectional commands.

**Table 127 — XDWRITEREAD (10) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 127 for the XDWRITEREAD (10) command.

See the WRITE (10) command (see 5.33) for the definitions of the WRPROTECT field, the FUA bit, and the FUA_NV bit.

See the READ (10) command (see 5.11) for the definition of the DPO bit.

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

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

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

If the XORPINFO bit is set to one, the device server supports protection information, and the medium has not been formatted with protection information, then the device server shall terminate the command with CHECK

CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

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

See the PRE-FETCH (10) command (see 5.8) for the definition of the LOGICAL BLOCK ADDRESS field.

See the PRE-FETCH (10) command (see 5.8) and 4.23 for the definition of the GROUP NUMBER field.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that the device server shall read, transfer from the Data-Out Buffer, and XOR into a buffer, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. If the specified LBA and the specified transfer length exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.6.4).

The CONTROL byte is defined in SAM-5.

### 5.48 XDWRITEREAD (32) command

The XDWRITEREAD (32) command (see table 128) requests that the device server perform the actions defined for the XDWRITEREAD (10) command (see 5.47).

**Table 128 — XDWRITEREAD (32) command**

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

The OPERATION CODE byte, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-4 and shall be set to the values shown in table 128 for the XDWRITEREAD (32) command.

See the XDWRITEREAD (10) command (see 5.47) for the definitions of the other fields in this command.

### 5.49 XPWRITE (10) command

The XPWRITE (10) command (see table 129) requests that the device server perform the following as an uninterrupted sequence of actions (see 4.27):

1) if required by the rules for caching (see 4.15), perform read medium operations on the logical block(s) referenced by the LBA(s) specified by this command;
2) transfer the specified number of logical blocks from the Data-Out Buffer;
3) perform an XOR operation on:
   A) the user data contained in the logical blocks read from the medium or cache; and
   B) the user data contained in the logical blocks transferred from the Data-Out Buffer;
   and
4) perform write medium operations on the resulting XOR data to the specified LBA(s) based on the rules of caching.

Logical block data for this command may include protection information, based on the XORPINFO bit and the medium format.

**Table 129 — XPWRITE (10) command**

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

The OPERATION CODE byte is defined in SPC-4 and shall be set to the value shown in table 129 for the XPWRITE (10) command.

See the READ (10) command (see 5.11) for the definition of the DPO bit. See the WRITE (10) command (see 5.33) for the definitions of the FUA bit and the FUA_NV bit. See the PRE-FETCH (10) command (see 5.8) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.8) and 4.23 for the definition of the GROUP NUMBER field.

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

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

If the XORPINFO bit is set to one, the device server supports protection information, and the medium has not been formatted with protection information, then the device server shall terminate the command with CHECK

CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

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

The TRANSFER LENGTH field specifies the number of contiguous logical blocks that shall be read, XORed with logical blocks transferred from the Data-Out Buffer, and written, starting with the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field. If the specified LBA and the specified transfer length exceed the capacity of the medium (see 4.5), then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.6.3).

The CONTROL byte is defined in SAM-5.

### 5.50 XPWRITE (32) command

The XPWRITE (32) command (see table 130) requests that the device server perform the actions defined for the XPWRITE (10) command (see 5.49).

**Table 130 — XPWRITE (32) command**

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

The OPERATION CODE byte, the ADDITIONAL CDB LENGTH field, and the SERVICE ACTION field are defined in SPC-4 and shall be set to the values shown in table 130 for the XPWRITE (32) command.

See the XPWRITE (10) command (see 5.49) for the definitions of the other fields in this command.

# 6 Parameters for direct-access block devices

## 6.1 Parameters for direct-access block devices introduction

Table 131 shows the parameters for direct-access block devices defined in clause 6 and a reference to the subclause where each parameter type is defined.

**Table 131 — Parameters for direct-access block devices**

| Parameter type | Reference |
|---|---|
| Address descriptors | 6.2 |
| Diagnostic parameters | 6.3 |
| Log parameters | 6.4 |
| Mode parameters | 6.5 |
| Vital product data (VPD) parameters | 6.6 |
| Copy manager parameters | 6.7 |

## 6.2 Address descriptors

### 6.2.1 Address descriptor overview

This subclause describes the address descriptors (see table 132) used for:

a) the FORMAT UNIT command (see 5.3);
b) the READ DEFECT DATA commands (see 5.17 and 5.18); and
c) the Translate Address Input diagnostic page (see 6.3.4) for the SEND DIAGNOSTIC command (see
d) SPC-4) and the Translate Address Output diagnostic page (see 6.3.5) for the RECEIVE DIAGNOSTIC RESULTS command (see SPC-4).

The format type of an address descriptor is:

a) specified in the DEFECT LIST FORMAT field in the CDB for the FORMAT UNIT command (see 5.3.1);
b) indicated in the DEFECT LIST FORMAT field in the READ DEFECT DATA parameter data (see 5.17.2 and 5.18.2);
c) specified and indicated in the SUPPLIED FORMAT field for the Translate Address diagnostic pages; or
d) specified and indicated in the TRANSLATE FORMAT field for the Translate Address diagnostic pages.

Table 132 defines the types of address descriptors.

**Table 132 — Address descriptors**

| Type | Description | Reference |
|---|---|---|
| 000b | Short block format address descriptor | 6.2.2 |
| 001b | Extended bytes from index format address descriptor [a] | 6.2.3 |
| 010b | Extended physical sector format address descriptor [a] | 6.2.4 |
| 011b | Long block format address descriptor | 6.2.5 |
| 100b | Bytes from index format address descriptor [a] | 6.2.6 |
| 101b | Physical sector format address descriptor [a] | 6.2.7 |
| 110b | Vendor specific | |
| 111b | Reserved | |

[a] This address descriptor format type is defined for direct access block devices using rotating media (see 4.3.2).

### 6.2.2 Short block format address descriptor

A format type of 000b specifies the short block format address descriptor defined in table 133.

**Table 133 — Short block format address descriptor (000b)**

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

For the FORMAT UNIT parameter list, the SHORT BLOCK ADDRESS field specifies a four-byte LBA. If the physical block containing the logical block referenced by the specified LBA contains additional logical blocks, then the device server may consider the LBAs of those additional logical blocks to also have been specified.

For the READ DEFECT DATA parameter data, the SHORT BLOCK ADDRESS field indicates a vendor specific four-byte value.

For the Translate Address diagnostic pages, the SHORT BLOCK ADDRESS field contains a four-byte LBA or a vendor specific four-byte value that is greater than the capacity of the medium.

### 6.2.3 Extended bytes from index address descriptor

A format type of 001b specifies the extended bytes from index format address descriptor defined in table 134. For the FORMAT UNIT parameter list and the READ DEFECT DATA parameter data, this address descriptor contains the location of a defect that:

a) is the length of one track;
b) is less than the length of a physical block; or
c) starts from one address descriptor and extends to the next address descriptor.

For the Translate Address diagnostic pages, this address descriptor contains the location of an LBA. For the Translate Address Output diagnostic page (see 6.2.5), if the SUPPLIED FORMAT field is set to 001b and the MADS bit in the ADDRESS TO TRANSLATE field is set to one, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

**Table 134 — Extended bytes from index format address descriptor (001b)**

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

The CYLINDER NUMBER field contains the cylinder number.

The HEAD NUMBER field contains the head number.

A multi-address descriptor start (MADS) bit set to one specifies that this address descriptor defines the beginning of a defect that spans multiple address descriptors. The defect may be a number of sequential

physical blocks on the same cylinder and head (i.e., a track) or may span a number of sequential tracks on the same head. A MADS bit set to zero specifies that:

a) this address descriptor defines the end of a defect if the previous address descriptor has the MADS bit set to one; or
b) this address descriptor defines a single track that contains one or more defects (i.e., the BYTES FROM INDEX field contains FFF_FFFFh) or a single defect (i.e., the BYTES FROM INDEX field does not contain FFF_FFFFh).

See 4.13.2 for valid combinations of two address descriptors that describe a defect.

The BYTES FROM INDEX field:

a) if not set to FFF_FFFFh, contains the number of bytes from the index (e.g., from the start of the track) to the location being described; or
b) if set to FFF_FFFFh, specifies or indicates that the entire track is being described.

More than one logical block may be described by this address descriptor.

Table 135 defines the order of the fields used for sorting extended bytes from index format address descriptors if the command using the address descriptors specifies sorting.

**Table 135 — Sorting order for extended bytes from index format address descriptors**

| Bit: | (MSB) 59 | ••• | 36 | 35 | ••• | 28 | 27 | ••• | (LSB) 0 |
|---|---|---|---|---|---|---|---|---|---|
| | CYLINDER NUMBER field | | | HEAD NUMBER field | | | BYTES FROM INDEX field | | |

### 6.2.4 Extended physical sector format address descriptor

A format type of 010b specifies the extended physical sector format address descriptor defined in table 136. For the FORMAT UNIT parameter list and the READ DEFECT DATA parameter data, this address descriptor contains the location of a defect that:

a) is the length of one track;
b) is less than the length of a physical block; or
c) starts from one address descriptor and extends to the next address descriptor.

For the Translate Address diagnostic pages, this address descriptor specifies the location of an LBA. For the Translate Address Output diagnostic page (see 6.2.5), if the SUPPLIED FORMAT field is set to 010b and the MADS bit in the ADDRESS TO TRANSLATE field is set to one, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

**Table 136 — Extended physical sector format address descriptor (010b)**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | CYLINDER NUMBER | | | | | | | |
| 2 | | | | | | | | (LSB) |
| 3 | HEAD NUMBER | | | | | | | |
| 4 | MADS | Reserved | | | (MSB) | | | |
| ••• | SECTOR NUMBER | | | | | | | |
| 7 | | | | | | | | (LSB) |

The CYLINDER NUMBER field contains the cylinder number.

The HEAD NUMBER field contains the head number.

A multi-address descriptor start (MADS) bit set to one specifies that this address descriptor defines the beginning of a defect that spans multiple address descriptors. The defect may span a number of sequential physical blocks on the same cylinder and head (i.e., a track) or may span a number of sequential tracks on the same head. A MADS bit set to zero specifies that:

a) this address descriptor defines the end of a defect if the previous address descriptor has the MADS bit set to one; or
b) this address descriptor defines a single track that contains one or more defects (i.e., the BYTES FROM INDEX field contains FFF_FFFFh) or a single defect (i.e., the BYTES FROM INDEX field does not contain FFF_FFFFh).

See 4.13.2 for valid combinations of two address descriptors that describe a defect.

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

The SECTOR NUMBER field:

a) if not set to FFF_FFFFh, contains the sector number of the location being described; or
b) if set to FFF_FFFFh, specifies or indicates that the entire track is being described.

More than one logical block may be described by this address descriptor.

Table 137 defines the order of the fields used for sorting extended physical sector format address descriptors if the command using the address descriptors specifies sorting.

**Table 137 — Sorting order for extended physical sector format address descriptors**

| **Bit:** | **(MSB) 59** | **•••** | **36** | **35** | **•••** | **28** | **27** | **•••** | **(LSB) 0** |
|---|---|---|---|---|---|---|---|---|---|
| | CYLINDER NUMBER field | | | HEAD NUMBER field | | | SECTOR NUMBER field | | |

### 6.2.5 Long block format address descriptor

A format type of 011b specifies the long block format address descriptor defined in table 138.

**Table 138 — Long block format address descriptor (011b)**

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

For the FORMAT UNIT parameter list, the LONG BLOCK ADDRESS field specifies an eight-byte LBA. If the physical block containing the logical block referenced by the specified LBA contains additional logical blocks, then the device server may consider the LBAs of those additional logical blocks to also have been specified.

For the READ DEFECT DATA parameter data, the LONG BLOCK ADDRESS field indicates a vendor specific eight-byte value.

For the Translate Address diagnostic pages, the LONG BLOCK ADDRESS field contains an eight-byte LBA or a vendor specific eight-byte value that is greater than the capacity of the medium.

### 6.2.6 Bytes from index format address descriptor

A format type of 100b specifies the bytes from index format address descriptor defined in table 139. This address descriptor contains the location of a track or an offset from the start of a track.

**Table 139 — Bytes from index format address descriptor (100b)**

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

The CYLINDER NUMBER field contains the cylinder number.

The HEAD NUMBER field contains the head number.

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

More than one logical block may be described by this address descriptor.

Table 140 defines the order of the fields used for sorting bytes from index format address descriptors if the command using the address descriptors specifies sorting.

**Table 140 — Sorting order for bytes from index format address descriptors**

| Bit: | (MSB) 63 | ••• | 40 | 39 | ••• | 32 | 31 | ••• | (LSB) 0 |
|---|---|---|---|---|---|---|---|---|---|
| | CYLINDER NUMBER field | | | HEAD NUMBER field | | | BYTES FROM INDEX field | | |

### 6.2.7 Physical sector format address descriptor

A format type of 101b specifies the physical sector format address descriptor defined in table 141. This address descriptor contains the location of a track or a sector.

**Table 141 — Physical sector format address descriptor (101b)**

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

The CYLINDER NUMBER field contains the cylinder number.

The HEAD NUMBER field contains the head number.

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

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

Table 142 defines the order of the fields used for sorting physical sector format address descriptors if the command using the address descriptors specifies sorting.

**Table 142 — Sorting order for physical sector format address descriptors**

| **Bit:** | **(MSB) 63** | **...** | **40** | **39** | **...** | **32** | **31** | **...** | **(LSB) 0** |
|---|---|---|---|---|---|---|---|---|---|
| | CYLINDER NUMBER field | | | HEAD NUMBER field | | | SECTOR NUMBER field | | |

### 6.3 Diagnostic parameters

#### 6.3.1 Diagnostic parameters overview

This subclause defines the pages and descriptors for diagnostic parameters used by direct-access block devices.

The diagnostic pages and their corresponding page codes for direct-access block devices are defined in table 143.

**Table 143 — Diagnostic page codes for direct-access block devices**

| **Diagnostic page name** | **Page code** | **Reference** |
|---|---|---|
| Diagnostic pages assigned by SPC-4 | 30h to 3Fh | SPC-4 |
| Rebuild Assist Input diagnostic page | 42h | 6.3.2 |
| Rebuild Assist Output diagnostic page | | 6.3.3 |
| SCSI enclosure services diagnostic pages | 01h to 2Fh | SES-2 |
| Supported Diagnostic Page diagnostic page | 00h | SPC-4 |
| Translate Address Input diagnostic page | 40h | 6.3.4 |
| Translate Address Output diagnostic page | | 6.3.5 |
| Obsolete | 41h | |
| Vendor specific diagnostic pages | 80h to FFh | |
| Reserved for this standard | 43h to 7Fh | |

### 6.3.2 Rebuild Assist Input diagnostic page

An application client sends a RECEIVE DIAGNOSTIC RESULTS command to retrieve a Rebuild Assist Input diagnostic page (see table 144), which provides information about whether the rebuild assist mode (see 4.20) is enabled or not and a device server's rebuild assist mode capabilities.

**Table 144 — Rebuild Assist Input diagnostic page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PAGE CODE (42h) | | | | | | | |
| 1 | Reserved | | | | | | | |
| 2 | (MSB) | | | | | | | |
| 3 | PAGE LENGTH (4 + (2 × n)) | | | | | | | (LSB) |
| 4 | Reserved | | | | | | | ENABLED |
| 5 | Reserved | | | | | | | |
| 6 | | | | | | | | |
| 7 | PHYSICAL ELEMENT LENGTH (n) | | | | | | | |
| 8 | DISABLED PHYSICAL ELEMENT MASK | | | | | | | |
| ••• | | | | | | | | |
| 7 + n | | | | | | | | |
| 8 + n | DISABLED PHYSICAL ELEMENT | | | | | | | |
| ••• | | | | | | | | |
| 7 + (2 × n) | | | | | | | | |

The PAGE CODE field and the PAGE LENGTH field are defined in SPC-4 and shall be set to the values defined in table 144.

An ENABLED bit set to one indicates that the rebuild assist mode is enabled. An ENABLED bit set to zero indicates that the rebuild assist mode is disabled.

The PHYSICAL ELEMENT LENGTH field indicates the length in bytes of the DISABLED PHYSICAL ELEMENT MASK field and the length in bytes of the DISABLED PHYSICAL ELEMENT field.

The bits in the DISABLED PHYSICAL ELEMENT MASK field indicate the bits in the DISABLED PHYSICAL ELEMENT field that are supported. Each bit set to one in the DISABLED PHYSICAL ELEMENT MASK field indicates that the corresponding bit in the DISABLED PHYSICAL ELEMENT field is supported and may be set to one in a Rebuild Assist Output diagnostic page sent with a SEND DIAGNOSTIC command.

The bits in the DISABLED PHYSICAL ELEMENT field indicate the physical elements that are disabled in this logical unit. Each bit set to one indicates that a physical element is disabled, and the device server shall report predicted read errors and predicted write errors for the associated group of LBAs.

### 6.3.3 Rebuild Assist Output diagnostic page

An application client sends a SEND DIAGNOSTIC command to send a Rebuild Assist Output diagnostic page (see table 145) that:

a) enables or disables rebuild assist mode (see 4.20.2); and/or
b) puts the logical unit in a simulated failure mode by disabling physical elements in conjunction with rebuild assist mode (see 4.20.5).

**Table 145 — Rebuild Assist Output diagnostic page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PAGE CODE (42h) | | | | | | | |
| 1 | Reserved | | | | | | | |
| 2 | (MSB) | | | | | | | |
| 3 | PAGE LENGTH (4 + (2 × n)) | | | | | | | (LSB) |
| 4 | Reserved | | | | | | | ENABLE |
| 5 | Reserved | | | | | | | |
| 6 | | | | | | | | |
| 7 | PHYSICAL ELEMENT LENGTH (n) | | | | | | | |
| 8 | DISABLED PHYSICAL ELEMENT MASK | | | | | | | |
| ••• | | | | | | | | |
| 7 + n | | | | | | | | |
| 8 + n | DISABLE PHYSICAL ELEMENT | | | | | | | |
| ••• | | | | | | | | |
| 7 + (2 × n) | | | | | | | | |

The PAGE CODE field and the PAGE LENGTH field are defined in SPC-4 and shall be set to the values defined in table 145.

An ENABLE bit set to one specifies that, after all fields in this diagnostic page have been validated:

a) a self-test of the physical elements in the logical unit may be performed; and
b) rebuild assist mode is enabled.

An ENABLE bit set to zero specifies that:

a) rebuild assist mode shall be disabled;
b) the other fields in this page shall be ignored; and
c) all physical elements shall be enabled (i.e., shall not simulate a predicted error response).

The PHYSICAL ELEMENT LENGTH field shall be set to the same value that is returned in the PHYSICAL ELEMENT LENGTH field in the Rebuild Assist Input diagnostic page.

If the PHYSICAL ELEMENT LENGTH field is not set to the same value that is returned in the PHYSICAL ELEMENT LENGTH field in the Rebuild Assist Input diagnostic page, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The device server shall ignore the DISABLED PHYSICAL ELEMENT MASK field.

Each bit in the DISABLE PHYSICAL ELEMENT field specifies a physical element that shall be disabled. A bit set to one in the DISABLE PHYSICAL ELEMENT field specifies that the device server shall respond to read commands and write commands specifying LBAs associated with that physical element as if the associated LBAs have predicted errors. A bit set to zero in the DISABLE PHYSICAL ELEMENT field specifies that the device server shall

not respond to read commands and write commands specifying LBAs associated with that physical element as if the associated LBAs do not have predicted errors. If the ENABLE bit is set to one, and the DISABLE PHYSICAL ELEMENT field specifies:

a) any bits set to one that are not supported by the logical unit;
b) all bits that are supported by the logical unit are set to one; or
c) setting to zero any bits that are set to one,

then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

### 6.3.4 Translate Address Input diagnostic page

Table 146 defines the Translate Address Input diagnostic page sent by a device server in response to a RECEIVE DIAGNOSTIC RESULTS command after the Translate Address Output diagnostic page (see 6.3.4) has been sent by an application client with the SEND DIAGNOSTIC command. If a Translate Address Output diagnostic page has not yet been processed, the results of a RECEIVE DIAGNOSTIC RESULTS command requesting this diagnostic page are vendor specific.

**Table 146 — Translate Address Input diagnostic page**

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

The PAGE CODE field is defined in SPC-4 and shall be set to the value shown in table 146.

The PAGE LENGTH field is defined in SPC-4.

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

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

The SUPPLIED FORMAT field contains the value from the SUPPLIED FORMAT field in the previous Translate Address Output diagnostic page (see 6.3.5).

A reserved area (RAREA) bit set to zero indicates that no part of the translated address falls within a reserved area of the medium. A RAREA bit set to one indicates that all or part of the translated address falls within a reserved area of the medium (e.g., speed tolerance gap, alternate sector, or vendor reserved area). If the

entire translated address falls within a reserved area, then the device server may not return a translated address.

An alternate sector (ALTSEC) bit set to zero indicates that no part of the translated address is located in an alternate sector of the medium or that the device server is unable to determine this information. An ALTSEC bit set to one indicates that the translated address is physically located in an alternate sector of the medium. If the device server is unable to determine if all or part of the translated address is located in an alternate sector, then the device server shall set this bit to zero.

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

The TRANSLATED FORMAT field contains the value from the TRANSLATE FORMAT field in the previous Translate Address Output diagnostic page (see 6.3.4).

The TRANSLATED ADDRESS field(s) contains the address(es) the device server translated from the address supplied by the application client in the previous Translate Address Output diagnostic page. Each field shall be in the format specified in the TRANSLATE FORMAT field. The formats are described in 6.2. If the short block format address descriptor is specified, then the first four bytes of the TRANSLATED ADDRESS field shall contain the short block format address descriptor and the last four bytes shall contain 0000_0000h.

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

### 6.3.5 Translate Address Output diagnostic page

The Translate Address diagnostic pages provides a method for an application client to have a device server translate an address descriptor (see 6.2) from one format to another. The address descriptor(s) to be translated are sent to the device server in the Translate Address Output diagnostic page with the SEND DIAGNOSTIC command and the results are returned to the application client in the Translate Address Input diagnostic page by the RECEIVE DIAGNOSTIC RESULTS command.

Table 147 defines the format of the Translate Address Output diagnostic page sent with the SEND DIAGNOSTIC command. The translated address is returned in the Translate Address Input diagnostic page (see 6.3.4).

**Table 147 — Translate Address Output diagnostic page**

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

The PAGE CODE field and PAGE LENGTH field are defined in SPC-4 and shall be set to the values shown in table 147.

The SUPPLIED FORMAT field specifies the format of the ADDRESS TO TRANSLATE field. Valid values for this field are defined in the DEFECT LIST FORMAT field of the FORMAT UNIT command (see 5.3). If the device server does not support the requested format, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The TRANSLATE FORMAT field specifies the format the device server shall use for the result of the address translation. Valid values for this field are defined in the DEFECT LIST FORMAT field of the FORMAT UNIT command. If the device server does not support the specified format, then the device server shall terminate the SEND DIAGNOSTIC command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The ADDRESS TO TRANSLATE field contains a single address descriptor which the application client is requesting the device server to translate. The format of this field depends on the value in the SUPPLIED FORMAT field. The formats are described in 6.2. If the short block format address descriptor is specified, then the first four bytes of the ADDRESS TO TRANSLATE field shall contain the short block format address descriptor and the last four bytes shall contain 0000_0000h.

## 6.4 Log parameters

### 6.4.1 Log parameters overview

#### 6.4.1.1 Summary of log pages

This subclause defines the log pages and their corresponding page codes and subpage codes for direct-access block devices (see table 148). See SPC-4 for a detailed description of logging operations.

**Table 148 — Log page codes and subpage codes for direct-access block devices**

| Log page name | Page code [a] | Subpage code [a] | Reference |
|---|---|---|---|
| Application Client log page | 0Fh | 00h | SPC-4 |
| ATA PASS-THROUGH Results | 16h | 00h | SAT-3 |
| Background Scan Results log page | 15h | 00h | 6.4.2 |
| Buffer Over-Run/Under-Run log page | 01h | 00h | SPC-4 |
| Format Status log page | 08h | 00h | 6.4.3 |
| Informational Exceptions log page | 2Fh | 00h | SPC-4 |
| Last n Deferred Errors Or Asynchronous Events log page | 0Bh | 00h | SPC-4 |
| Last n Error Events log page | 07h | 00h | SPC-4 |
| Logical Block Provisioning log page | 0Ch | 00h | 6.4.4 |
| Non-Medium Error log page | 06h | 00h | SPC-4 |
| Non-volatile Cache log page | 17h | 00h | 6.4.5 |
| Protocol-Specific Port log pages | 18h | 00h to FEh | SPC-4 |
| Read Error Counter log page | 03h | 00h | SPC-4 |
| Self-Test Results log page | 10h | 00h | SPC-4 |
| Solid State Media log page | 11h | 00h | 6.4.6 |
| Start-Stop Cycle Counter log page | 0Eh | 00h | SPC-4 |
| Supported Log Pages log page | 00h | FFh | SPC-4 |
| Supported Log Pages and Subpages log page | 00h | 00h | SPC-4 |
| Supported Subpages | 01h to 3Fh | FFh | SPC-4 |
| Temperature log page | 0Dh | 00h | SPC-4 |
| Verify Error Counter log page | 05h | 00h | SPC-4 |
| Write Error Counter log page | 02h | 00h | SPC-4 |
| Restricted | 09h | 00h | SPC-4 |
| Restricted | 0Ah | 00h | SPC-4 |
| Vendor specific | 30h to 3Eh | 00h to FEh | n/a |

[a] All page code and subpage code combinations not shown in this table are reserved for direct-access block devices.

#### 6.4.1.2 Setting and resetting log parameters

In a LOG SELECT command (see SPC-4), an application client may specify that:

a) all the parameters in a log page or pages are to be reset (i.e., the PCR bit set to one and the PARAMETER LIST LENGTH field is set to zero); or
b) individual parameters in log page are to be changed to specified new values (i.e., the PCR bit is set to zero and the PARAMETER LIST LENGTH field is not set to zero).

The device server processing of these requests depends on the log parameter that is being reset or changed, and is specified in the table that defines the log parameter using the keywords defined in table 155 (also see SPC-4).

**Table 149 — Keywords for resetting or changing log parameters**

| Keyword | Device server processing when: | |
|---|---|---|
| | **PCR bit is set to one** [a] | **PCR bit is set to zero** [b] |
| Always | Reset the log parameter. | Change the log parameter. |
| Reset Only | Reset the log parameter. | If any changes are requested in the parameter value field of the log parameter, then:<br>a) terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST; and<br>b) do not make any requested changes in any field in any log parameter in any log page |
| Never | Do not reset the log parameter; see the LOG SELECT command in SPC-4 for description of possible error conditions. | |

[a] If the PCR bit is set to one, and the PARAMETER LIST LENGTH field is not set to zero, then the device server shall terminate the LOG SELECT command (see SPC-4).
[b] If the PCR bit is set to zero, and the PARAMETER LIST LENGTH field is set to zero. then no log parameters are changed (see SPC-4).

### 6.4.2 Background Scan log page

#### 6.4.2.1 Overview

Using the format shown in table 151, the Background Scan log page (page code 15h). reports information about background pre-scan operations (see 4.24.2) and background medium scan operations (see 4.24.3) and any logical blocks where an error was detected during a background scan operation. The parameter codes for the Background Scan log page are listed in table 150.

**Table 150 — Background Scan log page parameter codes**

| Parameter code | Description | Resettable or Changeable [a] | Reference | Support required |
|---|---|---|---|---|
| 0000h | Background Scan Status | Never | 6.4.2.2 | Mandatory |
| 0001h to 0800h | Background Scan Results | Reset Only | 6.4.2.3 | Optional [b] |
| 8000h to AFFFh | Vendor specific | | n/a | Optional |
| All others | Reserved | | | |

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 6.4.1.2.
[b] If the Background Scan log page is supported, then at least one Background Scan Results log parameter shall be supported.

The Background Scan log page has the format shown in table 151.

**Table 151 — Background Scan log page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | DS (1) | SPF (0) | PAGE CODE (15h) | | | | | |
| 1 | SUBPAGE CODE (00h) | | | | | | | |
| 2 | (MSB) | PAGE LENGTH (n - 3) | | | | | | |
| 3 | | | | | | | | (LSB) |
| **Background scan parameters** | | | | | | | | |
| 4 | Background scan parameter [first] (see table 150) | | | | | | | |
| ••• | | | | | | | | |
| | ⋮ | | | | | | | |
| | Background scan parameter [last] (see table 150) | | | | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | |

The disable save (DS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are described in SPC-4.

The DS bit, the SPF bit, the PAGE CODE field, and the SUBPAGE CODE field shall be set to the values shown in table 151 for the Background Scan log page.

When the device server processes a LOG SELECT command with the PCR bit set to one (see SPC-4), the device server shall:

a) not change the values in the Background Scan Status log parameter; and
b) delete all Background Scan Results log parameters.

### 6.4.2.2 Background Scan Status log parameter

The Background Scan Status log parameter has the format shown in table 152.

**Table 152 — Background Scan Status log parameter format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | PARAMETER CODE (0000h) | | | | |
| 1 | | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (0Ch) | | | | | | | |
| 4 | (MSB) | | | ACCUMULATED POWER ON MINUTES | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | Reserved | | | | | | | |
| 9 | BACKGROUND SCAN STATUS | | | | | | | |
| 10 | (MSB) | | | NUMBER OF BACKGROUND SCANS PERFORMED | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | (MSB) | | | BACKGROUND SCAN PROGRESS | | | | |
| 13 | | | | | | | | (LSB) |
| 14 | (MSB) | | | NUMBER OF BACKGROUND MEDIUM SCANS PERFORMED | | | | |
| 15 | | | | | | | | (LSB) |

The PARAMETER CODE field is described in SPC-4 and shall be set to the value shown in table 152 for the Background Scan Status log parameter.

The DU bit, the TSD bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for the Background Scan Status log parameter shall be set for a binary format list log parameter as described in SPC-4.

The PARAMETER LENGTH field is described in SPC-4 and shall be set to the value shown in table 152 for the Background Scan Status log parameter.

The ACCUMULATED POWER ON MINUTES field indicates the number of minutes the device server has been powered on since manufacturing.

Table 153 defines the BACKGROUND SCANNING STATUS field.

**Table 153 — BACKGROUND SCAN STATUS field**

| Code | Description |
|---|---|
| 00h | No background scan operation is active. |
| 01h | A background medium scan operation is active. |
| 02h | A background pre-scan operation is active. |
| 03h | A background scan operation was halted due to a fatal error. |
| 04h | A background scan operation was halted due to a vendor specific pattern of errors. |
| 05h | A background scan operation was halted due to the medium being formatted without the P-list. |
| 06h | A background scan operation was halted due to a vendor specific cause. |
| 07h | A background scan operation was halted due to the temperature being out of the allowed range. |
| 08h | Background medium scan operations are enabled (i.e., the EN_BMS bit is set to one in the Background Control mode page (see 6.5.4)), and no background medium scan operation is active (i.e., the device server is waiting for Background Medium Scan Interval timer expiration before starting the next background medium scan operation). |
| 09h to FFh | Reserved |

The NUMBER OF BACKGROUND SCANS PERFORMED field indicates the number of background scan operations (i.e., the total number of background pre-scan operations plus the number of background medium scan operations) that have been performed since the SCSI target device was shipped by the manufacturer.

The BACKGROUND SCAN PROGRESS field indicates the percent complete of a background scan operation in progress. The returned value is a numerator that has 65 536 (i.e., 1_0000h) as its denominator. If there is no background scan operation in progress (i.e., no background scan operation has been initiated since power on or the most recent background scan operation has completed), then the device server shall set the BACKGROUND SCAN PROGRESS field to 0000h.

The NUMBER OF BACKGROUND MEDIUM SCANS PERFORMED field indicates the number of background medium scan operations that have been performed since the SCSI target device was shipped by the manufacturer. If the NUMBER OF BACKGROUND MEDIUM SCANS PERFORMED field contains 0000h, then the number of background medium scan operations is not reported.

The total number of background pre-scan operations that have been performed is the value in the NUMBER OF BACKGROUND SCANS PERFORMED field minus the value in the NUMBER OF BACKGROUND MEDIUM SCANS PERFORMED field.

### 6.4.2.3 Background Scan Results log parameter

A Background Scan Results log parameter has the format shown in table 154. If the Background Scan log page is reset, then all Background Scan Results log parameters are discarded. If no errors have occurred during a background scan or the Background Scan log page has been reset, then no Background Scan Results log parameters shall be present.

**Table 154 — Background Scan Results log parameter format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| 1 | PARAMETER CODE (0001h to 0800h) | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (14h) | | | | | | | |
| 4 | (MSB) | | | | | | | |
| ••• | ACCUMULATED POWER ON MINUTES | | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | REASSIGN STATUS | | | | SENSE KEY | | | |
| 9 | ADDITIONAL SENSE CODE | | | | | | | |
| 10 | ADDITIONAL SENSE CODE QUALIFIER | | | | | | | |
| 11 | | | | | | | | |
| ••• | Vendor specific | | | | | | | |
| 15 | | | | | | | | |
| 16 | (MSB) | | | | | | | |
| ••• | LOGICAL BLOCK ADDRESS | | | | | | | |
| 23 | | | | | | | | (LSB) |

The PARAMETER CODE field is described in SPC-4 and shall be set to a value from 0001h through 0800h in sequence as errors are discovered during a background scan operation. When all of the supported parameter code values have been used, and a new error is discovered during a background scan operation, the oldest Background Scan Results log parameter in the list (i.e., the Background Scan Results log parameter with the smallest value in the ACCUMULATED POWER ON MINUTES field) shall be discarded, and the PARAMETER CODE field in the Background Scan Results log parameter for the new defect shall be set to the parameter code value of the discarded Background Scan Results log parameter.

The DU bit, the TSD bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for a Background Scan Results log parameter shall be set for a binary format list log parameter as described in SPC-4.

The PARAMETER LENGTH field is defined in SPC-4 and shall be set to the value shown in table 154 for a Background Scan Results log parameter.

The ACCUMULATED POWER ON MINUTES field indicates the number of minutes that the device server has been powered on since manufacturing at the time the background scan error reported in the Background Scan Results log parameter occurred.

Table 155 defines the REASSIGN STATUS field.

**Table 155 — REASSIGN STATUS field**

| Code | LOWIR bit [a] | | Reason | |
|---|---|---|---|---|
| | 0 | 1 | Original error [b] | Additional conditions |
| 1h | Yes | Yes | Recovered or unrecovered | The LBA has not yet been reassigned. [c] |
| 2h | Yes | No | Recovered | The device server performed automatic read reassignment for the LBA (i.e., performed a reassign operation for the LBA and a write operation with recovered logical block data). [d] |
| 4h | Yes | Yes | Recovered | The device server's attempt to perform automatic read reassignment failed. The logical block may or may not now have an uncorrectable error. [c] |
| 5h | Yes | No | Recovered | The error was corrected by the device server rewriting the logical block without performing a reassign operation. |
| 6h | Yes | Yes | Recovered or unrecovered | Either:<br>a) an application client caused automatic write reassignment for the LBA with a command performing a write operation; or<br>b) the LBPRZ bit is set to one in the Logical Block Provisioning VPD page (see 6.6.4), and an application client caused an unmap operation for the LBA. [c] |
| 7h | Yes | Yes | Recovered or unrecovered | Either:<br>a) an application client caused a reassign operation for the LBA with a REASSIGN BLOCKS command; or<br>b) the LBPRZ bit is set to zero in the Logical Block Provisioning VPD page (see 6.6.4), and an application client caused an unmap operation for the LBA. [c] |
| 8h | Yes | Yes | Recovered or unrecovered | An application client's request for a reassign operation for the LBA with a REASSIGN BLOCKS command failed. The logical block referenced by the LBA may or may not still have an uncorrectable error. |
| All others | Reserved | | | |

[a] Based on the LOWIR bit in the Background Control mode page (see 6.5.4), "No" specifies that a Background Scan Results log parameter shall not be generated for the error and "Yes" specifies that a Background Scan Results log parameter shall be generated for the error.

[b] Type of error detected while reading the logical block referenced by the LBA specified by the LOGICAL BLOCK ADDRESS field during a background scan operation.

[c] The REASSIGN STATUS field in a given log parameter changes from 1h or 4h to 6h, 7h, or 8h when a reassign operation, write medium operation based on the rules for caching, or unmap operation for the LBA succeeds or when a reassign operation for the LBA fails. After the LBA is reassigned, any subsequent medium error occurring for the LBA is reported in a new log parameter with the same value in the LOGICAL BLOCK ADDRESS field as the value in the LOGICAL BLOCK ADDRESS field in the log parameter for the previous medium error for the LBA.

[d] The ARRE bit in the Read-Write Error Recovery mode page (see 6.5.8) controls automatic read reassignment based on errors detected during all read operations, including those that are part of background scan operations.

If sense data is available, then the device server shall set the SENSE KEY field, the ADDITIONAL SENSE CODE

field, and the ADDITIONAL SENSE CODE QUALIFIER field to a hierarchy of additional information relating to error conditions that occurred during the background scan operation. The content of these fields is represented in the same format used by the sense data (see SPC-4).

The LOGICAL BLOCK ADDRESS field indicates the LBA associated with the medium error.

### 6.4.3 Format Status log page

#### 6.4.3.1 Overview

Using format shown table 157, the Format Status log page (page code 08h) reports information about the most recent successful format operation and the state of the direct-access block device since that operation was performed. The parameter codes for the Format Status log page are listed in table 156.

**Table 156 — Format Status log page parameter codes**

| Parameter code | Description | Resettable or Changeable [a] | Reference | Support Required |
|---|---|---|---|---|
| 0000h | Format Data Out | Never | 6.4.3.2 | Mandatory |
| 0001h | Grown Defects During Certification | Never | 6.4.3.3 | Mandatory |
| 0002h | Total Blocks Reassigned During Format | Never | 6.4.3.4 | Mandatory |
| 0003h | Total New Blocks Reassigned | Never | 6.4.3.5 | Mandatory |
| 0004h | Power On Minutes Since Format | Never | 6.4.3.6 | Mandatory |
| 0005h to 7FFFh | Reserved | | | |
| 8000h to FFFFh | Vendor specific | | n/a | Optional |

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 6.4.1.2.

The Format Status log page has the format shown in table 157.

**Table 157 — Format Status log page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | DS (1) | SPF (0b) | PAGE CODE (08h) | | | | | |
| 1 | SUBPAGE CODE (00h) | | | | | | | |
| 2 | (MSB) | | | | | | | |
| 3 | PAGE LENGTH (n - 3) | | | | | | | (LSB) |
| **Format status log parameters** | | | | | | | | |
| 4 | Format status log parameter [first] (see table 156) | | | | | | | |
| ••• | | | | | | | | |
| | | | | | | | | |
| | ⋮ | | | | | | | |
| | Format status log parameter [last] (see table 156) | | | | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | |

The disable save (DS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are described in SPC-4.

The DS bit, the SPF bit, the PAGE CODE field, and the SUBPAGE CODE field shall be set to the values shown in table 157 for the Format Status log page.

If a format operation has never been performed by the logical unit, then the parameter data field for each Format Status log parameter listed in table 156 is not defined by this standard. When a device server begins a format operation, the device server shall set each byte of the parameter data field, if any, to FFh for each Format Status log parameter (e.g., if the PARAMETER LENGTH field is set to 02h, then the parameter data field is set to FFFFh).

If the most recent format operation failed or the information for a Format Status log parameter is not available, then the device server shall return FFh in each byte of the log parameter data field, if any, for the log parameter (e.g., if the PARAMETER LENGTH field is set to 04h in the log parameter, then the log parameter data field shall be set to FFFF_FFFFh). The device server shall set each log parameter data field to be a multiple of four bytes.

#### 6.4.3.2 Format Data Out log parameter

The Format Data Out log parameter has the format shown in table 158.

**Table 158 — Format Data Out log parameter format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | PARAMETER CODE (0000h) | | | | |
| 1 | | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (n - 3) | | | | | | | |
| 4 | (MSB) | | | FORMAT DATA OUT | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | (LSB) |

The PARAMETER CODE field is described in SPC-4 and shall be set to the value shown in table 158 for the Format Data Out log parameter.

The DU bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for the Format Data Out log parameter shall be set for a binary format list log parameter as described in SPC-4.

The target save disable (TSD) bit (see SPC-4) shall be set to zero for the Format Data Out log parameter, indicating that the logical unit implicitly saves the Format Data Out log parameter at vendor specific intervals.

The PARAMETER LENGTH field is described in SPC-4.

After a successful format operation, the FORMAT DATA OUT field contains the FORMAT UNIT parameter list (see 5.3.2).

#### 6.4.3.3 Grown Defects During Certification log parameter

The Grown Defects During Certification log parameter has the format shown in table 159.

**Table 159 — Grown Defects During Certification log parameter format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| 1 | PARAMETER CODE (0001h) | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (08h) | | | | | | | |
| 4 | (MSB) | | | | | | | |
| ••• | GROWN DEFECTS DURING CERTIFICATION | | | | | | | |
| 11 | | | | | | | | (LSB) |

The PARAMETER CODE field is described in SPC-4 and shall be set to the value shown in table 159 for the Grown Defects During Certification log parameter.

The DU bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for the Grown Defects During Certification log parameter shall be set for a binary format list log parameters described in SPC-4.

The target save disable (TSD) bit (see SPC-4) shall be set to zero for the Grown Defects During Certification log parameter, indicating that the logical unit implicitly saves the Grown Defects During Certification log parameter at vendor specific intervals.

The PARAMETER LENGTH field is described in SPC-4 and shall be set to the value shown in table 159 for the Grown Defects During Certification log parameter.

After a successful format operation during which certification was performed, the GROWN DEFECTS DURING CERTIFICATION field contains the number of defects detected as a result of performing the certification. The value in the GROWN DEFECTS DURING CERTIFICATION field count reflects only those defects detected and replaced during the successful format operation that were not already part of the PLIST or GLIST. The GROWN DEFECTS DURING CERTIFICATION field shall be eight bytes.

After a successful format operation during which certification was not performed, the GROWN DEFECTS DURING CERTIFICATION field contains zero.

#### 6.4.3.4 Total Blocks Reassigned During Format log parameter

The Total Blocks Reassigned During Format log parameter has the format shown in table 160.

**Table 160 — Total Blocks Reassigned During Format log parameter format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| 1 | PARAMETER CODE (0002h) | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (08h) | | | | | | | |
| 4 | (MSB) | | | | | | | |
| ••• | TOTAL BLOCKS REASSIGNED DURING FORMAT | | | | | | | |
| 11 | | | | | | | | (LSB) |

The PARAMETER CODE field is described in SPC-4 and shall be set to the value shown in table 160 for the Total Blocks Reassigned During Format log parameter.

The DU bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for the Total Blocks Reassigned During Format log parameter shall be set for a binary format list log parameters described in SPC-4.

The target save disable (TSD) bit (see SPC-4) shall be set to zero for the Total Blocks Reassigned During Format log parameter, indicating that the logical unit implicitly saves the Total Blocks Reassigned During Format log parameter at vendor specific intervals.

The PARAMETER LENGTH field is described in SPC-4 and shall be set to the value shown in table 160 for the Total Blocks Reassigned During Format log parameter.

The TOTAL BLOCKS REASSIGNED DURING FORMAT field contains the count of the total number of logical blocks that were reassigned during the most recent successful format operation. The TOTAL BLOCKS REASSIGNED DURING FORMAT field shall be eight bytes.

#### 6.4.3.5 Total New Blocks Reassigned log parameter

The Total New Blocks Reassigned log parameter has the format shown in table 161.

**Table 161 — Total New Blocks Reassigned log parameter format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| 1 | PARAMETER CODE (0003h) | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (08h) | | | | | | | |
| 4 | (MSB) | | | | | | | |
| ••• | TOTAL NEW BLOCKS REASSIGNED | | | | | | | |
| 11 | | | | | | | | (LSB) |

The PARAMETER CODE field is described in SPC-4 and shall be set to the value shown in table 161 for the Total New Blocks Reassigned During Format log parameter.

The DU bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for the Total New Blocks Reassigned During Format log parameter shall be set for a binary format list log parameters described in SPC-4.

The target save disable (TSD) bit (see SPC-4) shall be set to zero for the Total New Blocks Reassigned During Format log parameter, indicating that the logical unit implicitly saves the Total New Blocks Reassigned During Format log parameter at vendor specific intervals.

The PARAMETER LENGTH field is described in SPC-4 and shall be set to the value shown in table 161 for the Total New Blocks Reassigned During Format log parameter.

The TOTAL NEW BLOCKS REASSIGNED field contains a count of the total number of logical blocks that have been reassigned since the completion of the most recent successful format operation. The TOTAL NEW BLOCKS REASSIGNED field shall contain eight bytes.

#### 6.4.3.6 Power On Minutes Since Format log parameter

The Power On Minutes Since Format log parameter has the format shown in table 162.

**Table 162 — Total Power On Minutes Since Format log parameter format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | PARAMETER CODE (0004h) | | | | |
| 1 | | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (04h) | | | | | | | |
| 4 | (MSB) | | | | | | | |
| ••• | POWER ON MINUTES SINCE FORMAT | | | | | | | |
| 7 | | | | | | | | (LSB) |

The PARAMETER CODE field is described in SPC-4 and shall be set to the value shown in table 162 for the Power On Minutes Since Format log parameter.

The DU bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for the Power On Minutes Since Format log parameter shall be set for a binary format list log parameter as described in SPC-4.

The target save disable (TSD) bit (see SPC-4) shall be set to zero for the Power On Minutes Since Format log parameter, indicating that the logical unit implicitly saves the for the Power On Minutes Since Format log parameter at vendor specific intervals.

The PARAMETER LENGTH field is described in SPC-4 and shall be set to the value shown in table 162 for the Power On Minutes Since Format log parameter.

The POWER ON MINUTES SINCE FORMAT field contains the unsigned number of usage minutes (i.e., minutes with power applied regardless of power state) that have elapsed since the most recent successful format operation. The POWER ON MINUTES SINCE FORMAT field shall be four bytes.

### 6.4.4 Logical Block Provisioning log page

#### 6.4.4.1 Overview

Using the format shown in table 164, the Logical Block Provisioning log page (page code 0Ch) reports the logical block provisioning status of the logical unit. The parameter codes for the Logical Block Provisioning log page are listed in table 163.

**Table 163 — Logical Block Provisioning log parameters**

| Parameter code [a] | Description | Resettable or Changeable [b] | Reference | Support Required |
|---|---|---|---|---|
| Resources that are associated with thresholds (0000h to 00FFh) | | | | |
| 0000h | Reserved | | | |
| 0001h | Available LBA Mapping Resource Count | Never | 6.4.4.2 | Optional [c] |
| 0002h | Used LBA Mapping Resource Count | Never | 6.4.4.4 | |
| 0003h to 00FFh | Reserved | | | |
| Resources that are not associated with thresholds (0000h to 00FFh) | | | | |
| 0100h | De-duplicated LBA Resource Count | Never | 6.4.4.5 | |
| 0101h | Compressed LBA Resource Count | Never | 6.4.4.6 | |
| 0102h | Total Efficiency LBA Resource Count | Never | 6.4.4.7 | |
| 0103h to FFEFh | Reserved | | | |
| FFF0h to FFFFh | Vendor specific | | | |

[a] Parameter codes 0001h to 00FFh are coordinated with the THRESHOLD RESOURCE field in the threshold descriptor of the Logical Block Provisioning mode page (see 6.5.7).

[b] The keywords in this column – Always, Reset Only, and Never – are defined in 6.4.1.2.

[c] If this log page is supported, then at least one parameter shall be supported. A logical block provisioning log parameter in the range 0001h to 00FFh should be provided to report resource usage for each threshold resource for which a threshold descriptor in the Logical Block Provisioning mode page (see 6.5.9) is available.

The Logical Block Provisioning log page has the format shown in table 164.

**Table 164 — Logical Block Provisioning log page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | DS (1) | SPF (0) | PAGE CODE (0Ch) | | | | | |
| 1 | SUBPAGE CODE (00h) | | | | | | | |
| 2 | (MSB) | PAGE LENGTH (n - 3) | | | | | | |
| 3 | | | | | | | | (LSB) |
| **Logical block provisioning parameter list** | | | | | | | | |
| 4 | Logical block provisioning log parameter [first] (see table 163) | | | | | | | |
| ••• | | | | | | | | |
| | | | | | | | | |
| | ⋮ | | | | | | | |
| | Logical block provisioning log parameter [last] (see table 163) | | | | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | |

The disable save (DS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are described in SPC-4.

The DS bit, the SPF bit, the PAGE CODE field, and the SUBPAGE CODE field shall be set to the values shown in table 164 for the Logical Block Provisioning log page.

### 6.4.4.2 Available LBA Mapping Resource Count log parameter

#### 6.4.4.2.1 Overview

The Available LBA Mapping Resource Count log parameter has the format shown in table 165.

**Table 165 — Available LBA Mapping Resource Count log parameter format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | PARAMETER CODE (0001h) | | | | | | |
| 1 | | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (08h) | | | | | | | |
| 4 | (MSB) | RESOURCE COUNT | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | Reserved | | | | | | SCOPE | |
| 9 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 11 | | | | | | | | |

The PARAMETER CODE field is described in SPC-4 and shall be set to the value shown in table 165 for the Available LBA Mapping Resource Count log parameter.

The DU bit, the TSD bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for the Available LBA Mapping Resource Count shall be set for a binary format list log parameter as described in SPC-4.

The PARAMETER LENGTH field is described in SPC-4 and shall be set to the value shown in table 165 for the Available LBA Mapping Resource Count.

The RESOURCE COUNT field indicates an estimate of the number of available LBA mapping resources and is defined in 6.4.4.3.

The SCOPE field indicates the scope to which the RESOURCE COUNT field applies and is defined in table 166.

**Table 166 — SCOPE field**

| Code | Description |
|---|---|
| 00b | The scope of the resource count is not reported. |
| 01b | The RESOURCE COUNT field indicates a resource that is dedicated to the logical unit. Usage of resources on other logical units does not impact the resource count. |
| 10b | The RESOURCE COUNT field indicates a resource that is not dedicated to the logical unit. Usage of resources on other logical units may impact the resource count. |
| 11b | Reserved |

#### 6.4.4.3 RESOURCE COUNT field

The RESOURCE COUNT field indicates the number of LBA resources expressed as a number of threshold sets for the threshold resource indicated by the parameter code value. The nominal number of LBA resources is calculated as follows:

LBA resources = resource count × threshold set size

where:

resource count is the value in the RESOURCE COUNT field

threshold set size is the number of LBAs in each threshold set (i.e., $2^{(\text{threshold exponent})}$ indicated in the Logical Block Provisioning VPD page (see 6.6.5))

#### 6.4.4.4 Used LBA Mapping Resource Count log parameter

The Used LBA Mapping Resource Count log parameter has the format shown in table 167.

**Table 167 — Used LBA Mapping Resource Count log parameter format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| 1 | PARAMETER CODE (0002h) | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (08h) | | | | | | | |
| 4 | (MSB) | | | | | | | |
| ••• | RESOURCE COUNT | | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | Reserved | | | | | | SCOPE | |
| 9 | | | | | | | | |
| ••• | Reserved | | | | | | | |
| 11 | | | | | | | | |

The PARAMETER CODE field is described in SPC-4 and shall be set to the value shown in table 167 for the Used LBA Mapping Resource Count log parameter.

The DU bit, the TSD bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for the Used LBA Mapping Resource Count log parameter shall be set for a binary format list log parameter as described in SPC-4.

The PARAMETER LENGTH field is described in SPC-4 and shall be set to the value shown in table 167 for the Used LBA Mapping Resource Count log parameter.

The RESOURCE COUNT field indicates an estimate of the number of used LBA mapping resources and is defined in 6.4.4.3.

The SCOPE field indicates the scope to which the RESOURCE COUNT field applies and is defined in table 166.

### 6.4.4.5 De-duplicated Resource Count log parameter

The De-duplicated LBA Resource Count log parameter (see table 168) contains information about de-duplicated LBA resources.

**Table 168 — De-duplicated Resource Count log parameter format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| 1 | PARAMETER CODE (0100h) | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (08h) | | | | | | | |
| 4 | (MSB) | | | | | | | |
| ••• | RESOURCE COUNT | | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | Reserved | | | | | | SCOPE | |
| 9 | | | | | | | | |
| ••• | Reserved | | | | | | | |
| 11 | | | | | | | | |

The PARAMETER CODE field is described in SPC-4 and shall be set to the value shown in table 168 for the De-duplicated Resource Count log parameter.

The DU bit, the TSD bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for the De-duplicated Resource Count log parameter shall be set for a binary format list log parameter as described in SPC-4.

The PARAMETER LENGTH field is described in SPC-4 and shall be set to the value shown in table 168 for the De-duplicated Resource Count log parameter.

The RESOURCE COUNT field indicates an estimate of the number of LBA resources made available as a result of de-duplication and is defined in 6.4.4.3.

The SCOPE field indicates the scope to which the RESOURCE COUNT field applies and is defined in table 166.

### 6.4.4.6 Compressed LBA Resource Count log parameter

The Compressed LBA Resource Count log parameter (see table 169) contains information about compressed LBA resources.

**Table 169 — Compressed LBA Resource Count log parameter format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| 1 | PARAMETER CODE (0101h) | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (08h) | | | | | | | |
| 4 | (MSB) | | | | | | | |
| ••• | RESOURCE COUNT | | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | Reserved | | | | | | SCOPE | |
| 9 | | | | | | | | |
| ••• | Reserved | | | | | | | |
| 11 | | | | | | | | |

The PARAMETER CODE field is described in SPC-4 and shall be set to the value shown in table 169 for the Compressed LBA Resource Count log parameter.

The DU bit, the TSD bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for the Compressed LBA Resource Count log parameter shall be set for a binary format list log parameter as described in SPC-4.

The PARAMETER LENGTH field is described in SPC-4 and shall be set to the value shown in table 169 for the Compressed LBA Resource Count log parameter.

The RESOURCE COUNT field indicates an estimate of the number of LBA resources made available as a result of compression and is defined in 6.4.4.3.

The SCOPE field indicates the scope to which the RESOURCE COUNT field applies and is defined in table 166.

#### 6.4.4.7 Total Efficiency LBA Resource Count log parameter

The Total Efficiency LBA Resource Count log parameter (see table 170) contains information about the combined effects of all LBA resource efficiencies (e.g., the result of the combination of de-duplicated LBA resources and compressed LBA resources).

**Table 170 — Total Efficiency LBA Resource Count log parameter format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| 1 | PARAMETER CODE (0102h) | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (08h) | | | | | | | |
| 4 | (MSB) | | | | | | | |
| ••• | RESOURCE COUNT | | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | Reserved | | | | | | SCOPE | |
| 9 | | | | | | | | |
| ••• | Reserved | | | | | | | |
| 11 | | | | | | | | |

The PARAMETER CODE field is described in SPC-4 and shall be set to the value shown in table 170 for the Total Efficiency LBA Resource Count log parameter.

The DU bit, the TSD bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for the Total Efficiency LBA Resource Count log parameter shall be set for a binary format list log parameter as described in SPC-4.

The PARAMETER LENGTH field is described in SPC-4 and shall be set to the value shown in table 170 for the Total Efficiency LBA Resource Count log parameter.

The RESOURCE COUNT field indicates an estimate of the number of LBA resources made available by the combined effects of all LBA resource efficiency methods (e.g., de-duplication and compression) and is defined in 6.4.4.3. The algorithm used to calculate this value is not defined by this standard.

The SCOPE field indicates the scope to which the RESOURCE COUNT field applies and is defined in table 166.

### 6.4.5 Non-volatile Cache log page

#### 6.4.5.1 Overview

Using the format shown in table 172, the Nonvolatile Cache log page reports the status of battery backup for a nonvolatile cache. The parameter codes for the Nonvolatile Cache log page are listed in table 171.

**Table 171 — Nonvolatile Cache log parameters**

| Parameter code | Description | Resettable or Changeable [a] | Reference | Support Required |
|---|---|---|---|---|
| 0000h | Remaining Nonvolatile Time | Never | 6.4.5.2 | Mandatory |
| 0001h | Maximum Nonvolatile Time | Never | 6.4.5.3 | Mandatory |
| All others | Reserved | | | |

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 6.4.1.2.

The Nonvolatile Cache log page has the format shown in table 172.

**Table 172 — Nonvolatile Cache log page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | DS | SPF (0b) | PAGE CODE (17h) | | | | | |
| 1 | SUBPAGE CODE (00h) | | | | | | | |
| 2 | (MSB) | PAGE LENGTH (n - 3) | | | | | | |
| 3 | | | | | | | | (LSB) |
| **Nonvolatile cache log parameters** | | | | | | | | |
| 4 | Non-volatile cache log parameter [first] (see table 171) | | | | | | | |
| ••• | | | | | | | | |
| | | | | | | | | |
| | ⋮ | | | | | | | |
| | | | | | | | | |
| ••• | Nonvolatile cache log parameter [last] (see table 171) | | | | | | | |
| n | | | | | | | | |

The disable save (DS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are described in SPC-4.

The SPF bit, the PAGE CODE field, and the SUBPAGE CODE field shall be set to the values shown in table 172 for the Nonvolatile Cache log page.

### 6.4.5.2 Remaining Nonvolatile Time log parameter

The Remaining Nonvolatile Time log parameter has the format shown in table 173.

**Table 173 — Remaining Nonvolatile Time parameter data**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | PARAMETER CODE (0000h) | | | | | | |
| 1 | | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (04h) | | | | | | | |
| 4 | Obsolete | | | | | | | |
| 5 | (MSB) | REMAINING NONVOLATILE TIME | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | (LSB) |

The PARAMETER CODE field is described in SPC-4 and shall be set to the value shown in table 173 for the Remaining Nonvolatile Time log parameter.

The DU bit, the TSD bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for the Remaining Nonvolatile Time log parameter shall be set for a binary format list log parameter as described in SPC-4.

The PARAMETER LENGTH field is described in SPC-4 and shall be set to the value shown in table 173 for the Remaining Nonvolatile Time log parameter.

The REMAINING NONVOLATILE TIME field is defined in table 174.

**Table 174 — REMAINING NONVOLATILE TIME field**

| Code | Description |
|---|---|
| 00_0000h | Nonvolatile cache is volatile, either permanently or temporarily (e.g., if batteries need to be recharged). |
| 00_0001h | Nonvolatile cache is expected to remain nonvolatile for an unknown amount of time (e.g., if battery status is unknown) |
| 00_0002h to FF_FFFEh | Nonvolatile cache is expected to remain nonvolatile for the number of minutes indicated (e.g., for the life of the battery supplying power to random access memory). |
| FF_FFFFh | Nonvolatile cache is indefinitely nonvolatile. |

### 6.4.5.3 Maximum Nonvolatile Time log parameter

The Maximum Nonvolatile Time log parameter has the format shown in table 175.

**Table 175 — Maximum Nonvolatile Time parameter data**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | PARAMETER CODE (0000h) | | | | |
| 1 | | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (04h) | | | | | | | |
| 4 | Obsolete | | | | | | | |
| 5 | (MSB) | | | MAXIMUM NONVOLATILE TIME | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | (LSB) |

The PARAMETER CODE field is described in SPC-4 and shall be set to the value shown in table 175 for the Maximum Nonvolatile Time log parameter.

The DU bit, the TSD bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for the Maximum Nonvolatile Time log parameter shall be set for a binary format list log parameter as described in SPC-4.

The PARAMETER LENGTH field is described in SPC-4 and shall be set to the value shown in table 175 for the Maximum Nonvolatile Time log parameter.

The MAXIMUM NONVOLATILE TIME field is defined in table 176.

**Table 176 — MAXIMUM NONVOLATILE TIME field**

| Code | Description |
|---|---|
| 00_0000h | Nonvolatile cache is volatile |
| 00_0001h | Reserved |
| 00_0002h to FF_FFFEh | Nonvolatile cache is capable of being nonvolatile for the estimated number of minutes indicated. If the time is based on batteries, then the time shall be based on the last full charge capacity rather than the design capacity of the batteries. |
| FF_FFFFh | Nonvolatile cache is indefinitely nonvolatile. |

### 6.4.6 Solid State Media log page

#### 6.4.6.1 Overview

Using the format shown in table 178, the Solid State media log page (page code 11h) reports parameters that are specific to SCSI target devices that contain solid state media. The parameter codes for the Solid State Media log page are listed in table 177.

**Table 177 — Solid State Media log parameters**

| Parameter code | Description | Resettable or Changeable [a] | Reference | Support Required |
|---|---|---|---|---|
| 0001h | Percentage Used Endurance Indicator | Never | 6.4.6.2 | Mandatory |
| All others | Reserved | | | |

[a] The keywords in this column – Always, Reset Only, and Never – are defined in 6.4.1.2.

The Solid State Media log page has the format shown in table 178.

**Table 178 — Solid State Media log page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | DS | SPF (0) | PAGE CODE (11h) | | | | | |
| 1 | SUBPAGE CODE (00h) | | | | | | | |
| 2 | (MSB) | PAGE LENGTH (n - 3) | | | | | | |
| 3 | | | | | | | | (LSB) |
| **Solid state media log parameters** | | | | | | | | |
| 4 | Solid state media parameter [first] (see table 177) | | | | | | | |
| ••• | | | | | | | | |
| | | | | | | | | |
| | ⋮ | | | | | | | |
| | | | | | | | | |
| ••• | Solid state media parameter [last] (see table 177) | | | | | | | |
| n | | | | | | | | |

The disable save (DS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are described in SPC-4.

The SPF bit, the PAGE CODE field, and the SUBPAGE CODE field shall be set to the values shown in table 178 for the Solid State Media log page.

### 6.4.6.2 Percentage Used Endurance Indicator log parameter

The Percentage Used Endurance Indicator log parameter has the format shown in table 179.

**Table 179 — Percentage Used Endurance Indicator log parameter format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| 1 | PARAMETER CODE (0001h) | | | | | | | (LSB) |
| 2 | Parameter control byte – binary format list log parameter (see SPC-4) | | | | | | | |
| | DU | Obsolete | TSD | ETC | TMC | | FORMAT AND LINKING | |
| 3 | PARAMETER LENGTH (04h) | | | | | | | |
| 4 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 6 | | | | | | | | |
| 7 | PERCENTAGE USED ENDURANCE INDICATOR | | | | | | | |

The PARAMETER CODE field is described in SPC-4 and shall be set to the value shown in table 179 for the Percentage Used Endurance Indicator log parameter.

The DU bit, the TSD bit, the ETC bit, the TMC field, and the FORMAT AND LINKING field for the Percentage Used Endurance Indicator log parameter shall be set for a binary format list log parameter as described in SPC-4.

The PARAMETER LENGTH field is described in SPC-4 and shall be set to the value shown in table 179 for the Percentage Used Endurance Indicator log parameter.

The PERCENTAGE USED ENDURANCE INDICATOR field indicates an estimate of the percentage of device life that has been used. The value in the field shall be set to zero at the time of manufacture. A value of 100 indicates that the estimated endurance of the device has been consumed, but may not indicate a device failure (e.g., minimum power-off data retention capability reached for devices using flash technology). The value is allowed to exceed 100. Values greater than 254 shall be reported as 255. The device server shall update the value at least once per power-on hour.

## 6.5 Mode parameters

### 6.5.1 Mode parameters overview

This subclause defines the mode pages and block descriptors (see 6.5.2) used by direct-access block devices. The mode pages and their corresponding page codes and subpage codes for direct-access block devices are shown in table 180.

**Table 180 — Mode page codes and subpage codes for direct-access block devices**

| Mode page name | Page code | Subpage code | Reference |
|---|---|---|---|
| Application Tag mode page | 0Ah | 02h | 6.5.3 |
| ATA Power Condition | 1Ah | F1h | SAT-3 |
| Background Control mode page | 1Ch | 01h | 6.5.4 |
| Caching mode page | 08h | 00h | 6.5.5 |
| Control mode page | 0Ah | 01h | SPC-4 |
| Control Extension mode page | 0Ah | 00h | SPC-4 |
| Disconnect-Reconnect mode page | 02h | 00h | SPC-4 |
| Enclosure Services Management mode page [a] | 14h | 00h | SES-2 |
| Informational Exceptions Control mode page | 1Ch | 00h | 6.5.6 |
| Logical Block Provisioning mode page | 1Ch | 02h | 6.5.7 |
| PATA Control | 0Ah | F1h | SAT-3 |
| Power Condition mode page | 1Ah | 00h | SPC-4 |
| Power Consumption mode page | 1Ah | 01h | SPC-4 |
| Protocol-Specific Logical Unit mode page | 18h | 00h | SPC-4 |
| Protocol-Specific Port mode page | 19h | 00h | SPC-4 |
| Read-Write Error Recovery mode page | 01h | 00h | 6.5.8 |
| Return all mode pages and subpages [b] | 3Fh | FFh | SPC-4 |
| Return all mode pages only (i.e., not including subpages) [b] | 3Fh | 00h | SPC-4 |
| Return all subpages for the specified mode page code [b] | 00h to 3Eh | FFh | SPC-4 |
| Verify Error Recovery mode page | 07h | 00h | 6.5.9 |
| Obsolete [c] | | | |
| Vendor specific [d] | | | |
| Reserved | all other page and subpage code combinations for direct-access block devices | | |

Note: SPC-4 contains a listing of mode page and subpage codes in numeric order.

[a] Valid only if the ENCSERV bit is set to one in the standard INQUIRY data (see SPC-4).
[b] Valid only for the MODE SENSE command.
[c] The following mode page codes are obsolete: 03h, 04h, 05h, 09h, 0Bh, 0Ch, 0Dh, and 10h.
[d] The following mode page code and subpage code combinations are vendor specific and do not require a page format: mode page code 00h with subpage code 00h and mode page codes 20h to 3Eh with all subpage codes.

The mode parameter list, including the mode parameter header, is described in SPC-4. Direct-access block devices support zero or one mode parameter block descriptors (i.e., the block descriptor is shared by all the logical blocks on the medium) (see 6.5.2).

The MEDIUM TYPE field in the mode parameter header (see SPC-4) shall be set to 00h.

The DEVICE-SPECIFIC PARAMETER field in the mode parameter header (see SPC-4) is defined for direct-access block devices in table 181.

**Table 181 — DEVICE-SPECIFIC PARAMETER field for direct-access block devices**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WP | Reserved | | DPOFUA | Reserved | | | |

If the medium is write-protected (i.e., due to mechanisms outside the scope of this standard), or the software write protect (SWP) bit in the Control mode page (see SPC-4) is set to one, then the device server shall set the WP bit to one when returning a DEVICE-SPECIFIC PARAMETER field in response to a MODE SENSE command. If the medium is not write-protected and the SWP bit is set to zero, then the device server shall set the WP bit to zero when returning a DEVICE-SPECIFIC PARAMETER field in response to a MODE SENSE command.

The write protect (WP) bit is not defined for mode data sent with a MODE SELECT command.

The DPOFUA bit is reserved for mode data sent with a MODE SELECT command.

If the device server does not support the DPO bit and the FUA bit (see 4.15) being set to one, then the device server shall set the DPO and FUA supported (DPOFUA) bit to zero when returning a DEVICE-SPECIFIC PARAMETER field in response to a MODE SENSE command. If the device server supports the DPO bit and the FUA bit being set to one, then the device server shall set the DPOFUA bit to one when returning a DEVICE-SPECIFIC PARAMETER field in response to a MODE SENSE command.

### 6.5.2 Mode parameter block descriptors

#### 6.5.2.1 Mode parameter block descriptors overview

If a device server returns a mode parameter block descriptor, then the device server shall return a short LBA mode parameter block descriptor (see 6.5.2.2) in the mode parameter data in response to:

a) a MODE SENSE (6) command; or
b) a MODE SENSE (10) command with the LLBAA bit set to zero.

If a device server returns a mode parameter block descriptor and the number of logical blocks is greater than FFFF_FFFFh, then the device server may return a long LBA mode parameter block descriptor (see 6.5.2.3) in the mode parameter data in response to a MODE SENSE (10) command with the LLBAA bit set to one.

If an application client sends a mode parameter block descriptor in the mode parameter list, then the application client shall send a short LBA mode parameter block descriptor (see 6.5.2.2) for a MODE SELECT (6) command.

If an application client sends a mode parameter block descriptor in the mode parameter list, then the application client may send a long LBA mode parameter block descriptor (see 6.5.2.3) for a MODE SELECT (10) command.

Support for the mode parameter block descriptors is optional. The device server shall establish a unit attention condition with the additional sense code set to MODE PARAMETERS CHANGED (see SPC-4 and SAM-5) when the block descriptor values are changed.

#### 6.5.2.2 Short LBA mode parameter block descriptor

Table 182 defines the short LBA mode parameter block descriptor for direct-access block devices used:

a) with the MODE SELECT (6) and MODE SENSE (6) commands; and
b) with the MODE SELECT (10) and MODE SENSE (10) commands when the LONGLBA bit is set to zero in the mode parameter header (see SPC-4).

**Table 182 — Short LBA mode parameter block descriptor**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | NUMBER OF LOGICAL BLOCKS | | | | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | Reserved | | | | | | | |
| 5 | (MSB) | | | | | | | |
| ••• | LOGICAL BLOCK LENGTH | | | | | | | |
| 7 | | | | | | | | (LSB) |

A device server shall respond to a MODE SENSE command (see SPC-4) by reporting the number of logical blocks specified in the NUMBER OF LOGICAL BLOCKS field sent in the last MODE SELECT command that contained a mode parameter block descriptor. If no MODE SELECT command with a mode parameter block descriptor has been received then the current number of logical blocks shall be returned. To determine the number of logical blocks at which the logical unit is currently formatted, the application client shall use a READ CAPACITY command (see 5.15 and 5.16) rather than the MODE SENSE command.

In response to a MODE SENSE command, the device server may return a value of zero indicating that it does not report the number of logical blocks in the short LBA mode parameter block descriptor.

In response to a MODE SENSE command, if the number of logical blocks on the medium exceeds the maximum value that is able to be specified in the NUMBER OF LOGICAL BLOCKS field, then the device server shall return a value of FFFF_FFFFh.

If the logical unit does not support changing its capacity by changing the NUMBER OF LOGICAL BLOCKS field using the MODE SELECT command (see SPC-4), then the value in the NUMBER OF LOGICAL BLOCKS field is ignored. If the device supports changing its capacity by changing the NUMBER OF LOGICAL BLOCKS field, then the NUMBER OF LOGICAL BLOCKS field is interpreted as follows:

a) if the NUMBER OF LOGICAL BLOCKS field is set to zero, then the logical unit shall retain its current capacity if the logical block length has not changed. If the NUMBER OF LOGICAL BLOCKS field is set to zero and the content of the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length, then the logical unit shall be set to its maximum capacity when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command);
b) if the NUMBER OF LOGICAL BLOCKS field is greater than zero and less than or equal to its maximum capacity, then the logical unit shall be set to that number of logical blocks. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length, then the logical unit shall not become format corrupt (see 4.10). This capacity setting shall be retained through power cycles, hard resets, logical unit resets, and I_T nexus losses. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length, then this capacity setting shall take effect on successful completion of the MODE SELECT command. If the content of the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length, then this capacity setting shall take effect when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command);
c) if the NUMBER OF LOGICAL BLOCKS field is set to a value greater than the maximum capacity of the device and less than FFFF_FFFFh, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the

additional sense code set to INVALID FIELD IN PARAMETER LIST. The logical unit shall retain its previous logical block descriptor settings; or

d) if the NUMBER OF LOGICAL BLOCKS field is set to FFFF_FFFFh, then the logical unit shall be set to its maximum capacity. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length, then the logical unit shall not become format corrupt (see 4.10). This capacity setting shall be retained through power cycles, hard resets, logical unit resets, and I_T nexus losses. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length, then this capacity setting shall take effect on successful completion of the MODE SELECT command. If the content of the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length, then this capacity setting shall take effect when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command).

If the device server supports changing its logical unit's capacity by changing the NUMBER OF LOGICAL BLOCKS field, is in a logical unit that supports logical block provisioning management, and the capacity is increased; then the additional LBAs shall be in the initial provisioning management condition as specified in 4.7.3.2 or 4.7.3.3.

The LOGICAL BLOCK LENGTH field specifies the length in bytes of each logical block. No change shall be made to any logical blocks on the medium until a format operation (see 5.2) is initiated by an application client.

A device server shall respond to a MODE SENSE command (see SPC-4) by reporting the length of the logical blocks as specified in the LOGICAL BLOCK LENGTH field sent in the last MODE SELECT command that contained a mode parameter block descriptor. If no MODE SELECT command with a block descriptor has been received then the current logical block length shall be returned (e.g., if the logical block length is 512 bytes and a MODE SELECT command occurs with the LOGICAL BLOCK LENGTH field set to 520 bytes, any MODE SENSE commands would return 520 in the LOGICAL BLOCK LENGTH field). To determine the logical block length at which the logical unit is currently formatted, the application client shall use a READ CAPACITY command (see 5.15 and 5.16) rather than a MODE SENSE command.

### 6.5.2.3 Long LBA mode parameter block descriptor

Table 183 defines the log LBA mode parameter block descriptor for direct-access block devices used with the MODE SELECT (10) command and MODE SENSE (10) command when the LONGLBA bit is set to one in the mode parameter header (see SPC-4).

**Table 183 — Long LBA mode parameter block descriptor**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | | | | NUMBER OF LOGICAL BLOCKS | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | | | | | | | | |
| ••• | | | | Reserved | | | | |
| 11 | | | | | | | | |
| 12 | (MSB) | | | | | | | |
| ••• | | | | LOGICAL BLOCK LENGTH | | | | |
| 15 | | | | | | | | (LSB) |

A device server shall respond to a MODE SENSE command (see SPC-4) by reporting the number of logical blocks specified in the NUMBER OF LOGICAL BLOCKS field sent in the last MODE SELECT command that contained a mode parameter block descriptor. If no MODE SELECT command with a mode parameter block descriptor has been received then the current number of logical blocks shall be returned. To determine the number of logical blocks at which the logical unit is currently formatted, the application client shall use a READ CAPACITY command (see 5.15 and 5.16) rather than a MODE SENSE command.

In response to a MODE SENSE command, the device server may return a value of zero indicating that it does not report the number of logical blocks in the long LBA mode parameter block descriptor.

If the logical unit does not support changing its capacity by changing the NUMBER OF LOGICAL BLOCKS field using the MODE SELECT command (see SPC-4), then the value in the NUMBER OF LOGICAL BLOCKS field is ignored. If the device supports changing its capacity by changing the NUMBER OF LOGICAL BLOCKS field, then the NUMBER OF LOGICAL BLOCKS field is interpreted as follows:

a) if the NUMBER OF LOGICAL BLOCKS field is set to zero, then the logical unit shall retain its current capacity if the logical block length has not changed. If the NUMBER OF LOGICAL BLOCKS field is set to zero and the content of the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length, then the logical unit shall be set to its maximum capacity when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command);
b) if the NUMBER OF LOGICAL BLOCKS field is greater than zero and less than or equal to its maximum capacity, then the logical unit shall be set to that number of logical blocks. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length, then the logical unit shall not become format corrupt (see 4.10). This capacity setting shall be retained through power cycles, hard resets, logical unit resets, and I_T nexus losses. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length, then this capacity setting shall take effect on successful completion of the MODE SELECT command. If the content of the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length, then this capacity setting shall take effect when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command);
c) if the NUMBER OF LOGICAL BLOCKS field is set to a value greater than the maximum capacity of the device and less than FFFF_FFFF_FFFF_FFFFh, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. The logical unit shall retain its previous block descriptor settings; or
d) if the NUMBER OF LOGICAL BLOCKS field is set to FFFF_FFFF_FFFF_FFFFh, then the logical unit shall be set to its maximum capacity. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length, then the logical unit shall not become format corrupt (see 4.10). This capacity setting shall be retained through power cycles, hard resets, logical unit resets, and I_T nexus losses. If the content of the LOGICAL BLOCK LENGTH field is the same as the current logical block length, then this capacity setting shall take effect on successful completion of the MODE SELECT command. If the content of the LOGICAL BLOCK LENGTH field (i.e., new logical block length) is different than the current logical block length, then this capacity setting shall take effect when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command).

If the device server supports changing its logical unit's capacity by changing the NUMBER OF LOGICAL BLOCKS field, supports logical block provisioning management, and the capacity is increased, then the additional LBAs shall be in the initial provisioning management condition as specified in 4.7.3.2 or 4.7.3.3.

The LOGICAL BLOCK LENGTH field specifies the length in bytes of each logical block. No change shall be made to any logical blocks on the medium until a format operation (see 5.2) is initiated by an application client.

A device server shall respond to a MODE SENSE command (see SPC-4) by reporting the length of the logical blocks as specified in the LOGICAL BLOCK LENGTH field sent in the last MODE SELECT command that contained a mode parameter block descriptor. If no MODE SELECT command with a block descriptor has been received then the current logical block length shall be returned (e.g., if the logical block length is 512 bytes and a MODE SELECT command occurs with the LOGICAL BLOCK LENGTH field set to 520 bytes, any MODE SENSE commands would return 520 in the LOGICAL BLOCK LENGTH field). To determine the logical block length at which the logical unit is currently formatted, the application client shall use a READ CAPACITY command (see 5.15 and 5.16) rather than a MODE SELECT command.

### 6.5.3 Application Tag mode page

#### 6.5.3.1 Introduction

The Application Tag mode page (see table 184) specifies the Application Tag that a device server configured for protection information (see 4.22.2) shall use for each LBA range if the ATO bit in the Control mode page (see SPC-4) is set to one. The mode page policy (see SPC-4) for this page shall be shared.

If a method not defined by this standard changes the parameter data to be returned by the device server in the Application Tag mode page, then the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus with the additional sense code set to MODE PARAMETERS CHANGED.

**Table 184 — Application Tag mode page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS | SPF (1b) | PAGE CODE (0Ah) | | | | | |
| 1 | SUBPAGE CODE (02h) | | | | | | | |
| 2 | (MSB) | | | | | | | |
| 3 | PAGE LENGTH (n - 3) | | | | | | | (LSB) |
| 4 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |
| **Application Tag descriptors** | | | | | | | | |
| 16 | Application Tag descriptor [first] (see 6.5.3.2) | | | | | | | |
| ••• | | | | | | | | |
| 39 | | | | | | | | |
| | ⋮ | | | | | | | |
| n - 24 | Application Tag descriptor [last] (see 6.5.3.2) | | | | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | |

The parameters saveable (PS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are defined in SPC-4.

The SPF bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field shall be set to the values shown in table 184.

The application tag descriptors are defined in 6.5.3.2.

### 6.5.3.2 Application Tag descriptor

The Application Tag descriptor format is described in table 185.

**Table 185 — Application Tag descriptor format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | LAST | Reserved | | | | | | |
| 1 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 5 | | | | | | | | |
| 6 | (MSB) | LOGICAL BLOCK APPLICATION TAG | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | (MSB) | LOGICAL BLOCK ADDRESS | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | (LSB) |
| 16 | (MSB) | LOGICAL BLOCK COUNT | | | | | | |
| ••• | | | | | | | | |
| 23 | | | | | | | | (LSB) |

A LAST bit set to one specifies that this Application Tag descriptor is the last valid Application Tag descriptor in the Application Tag mode page. A LAST bit set to zero specifies that the Application Tag descriptor is not the last valid Application Tag descriptor in the Application Tag mode page.

The LOGICAL BLOCK APPLICATION TAG field specifies the value to be compared with the LOGICAL BLOCK APPLICATION TAG field associated with data read or written to the LBA.

The LOGICAL BLOCK ADDRESS field contains the starting LBA for this Application Tag descriptor. The LOGICAL BLOCK ADDRESS field in the first Application Tag descriptor shall be set to 0000_0000_0000_0000h. For subsequent Application Tag descriptors, the contents of the LOGICAL BLOCK ADDRESS field shall contain the sum of the values in:

a) the LOGICAL BLOCK ADDRESS field in the previous Application Tag descriptor; and
b) the LOGICAL BLOCK COUNT field in the previous Application Tag descriptor.

The sum of the LOGICAL BLOCK ADDRESS field in the Application Tag descriptor with the LAST bit set to one and the LOGICAL BLOCK COUNT field in the Application Tag descriptor with the LAST bit set to one shall equal the RETURNED LOGICAL BLOCK ADDRESS field in the READ CAPACITY (16) parameter data (see 5.16.2).

If an invalid combination of the LAST bit, LOGICAL BLOCK APPLICATION TAG field, and LOGICAL BLOCK ADDRESS field are sent by the application client, then the device server shall terminate the MODE SELECT command (see SPC-4) with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The LOGICAL BLOCK COUNT field specifies the number of logical blocks to which this Application Tag descriptor applies.

A LOGICAL BLOCK COUNT field set to 0000_0000_0000_0000h specifies that this Application Tag descriptor shall be ignored.

### 6.5.4 Background Control mode page

The Background Control mode page (see table 186) provides controls over background scan operations (see 4.24). The mode page policy (see SPC-4) for this subpage shall be shared.

**Table 186 — Background Control mode page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS | SPF (1b) | PAGE CODE (1Ch) | | | | | |
| 1 | SUBPAGE CODE (01h) | | | | | | | |
| 2 | (MSB) | PAGE LENGTH (000Ch) | | | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | Reserved | | | | | S_L_FULL | LOWIR | EN_BMS |
| 5 | Reserved | | | | | | | EN_PS |
| 6 | (MSB) | BACKGROUND MEDIUM SCAN INTERVAL TIME | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | (MSB) | BACKGROUND PRE-SCAN TIME LIMIT | | | | | | |
| 9 | | | | | | | | (LSB) |
| 10 | (MSB) | MINIMUM IDLE TIME BEFORE BACKGROUND SCAN | | | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | (MSB) | MAXIMUM TIME TO SUSPEND BACKGROUND SCAN | | | | | | |
| 13 | | | | | | | | (LSB) |
| 14 | Reserved | | | | | | | |
| 15 | | | | | | | | |

The parameters saveable (PS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are defined in SPC-4.

The SPF bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field shall be set to the values shown in table 186.

A suspend on log full (S_L_FULL) bit set to zero specifies that the device server shall continue running a background scan operation (see 4.24.1) even if the Background Scan Results log page (see 6.4.2) contains the maximum number of Background Scan log parameters (see 6.4.2.3) supported by the logical unit. A S_L_FULL bit set to one specifies that the device server shall suspend a background scan operation if the Background Scan Results log page contains the maximum number of Background scan log parameters supported by the logical unit.

A log only when intervention required (LOWIR) bit set to zero specifies that the device server shall log all suspected recoverable medium errors or unrecoverable medium errors that are identified during background scan operations in the Background Scan Results log page. A LOWIR bit set to one specifies that the device server shall only log medium errors identified during back ground scan operations in the Background Scan Results log page that require application client intervention.

An enable background medium scan (EN_BMS) bit set to zero specifies that the device server shall disable background medium scan operations (see 4.24.3). An EN_BMS bit set to one specifies that the device server shall enable background medium scan operations. If the EN_PS bit is also set to one, and a background pre-scan operation is in progress, then the logical unit shall not start a background medium scan operation until after the background pre-scan operation is halted or completed. If a background medium scan operation is in progress when the EN_BMS bit is changed from one to zero, then the logical unit shall suspend the

background medium scan operation before the device server completes the MODE SELECT command, and the background medium scan shall remain suspended until the EN_BMS bit is set to one, at which time the logical unit shall resume the background medium scan operation beginning with the logical block being tested when the background medium scan operation was suspended.

An enable pre-scan (EN_PS) bit set to zero specifies that the device server shall disable background pre-scan operations (see 4.24.2). If a background pre-scan operation is in progress when the EN_PS bit is changed from a one to a zero, then the logical unit shall halt the background pre-scan operation before the device server completes the MODE SELECT command. An EN_PS bit set to one specifies that the logical unit shall start a background pre-scan operation after the next power on. Once a logical unit has completed a background pre-scan operation, the logical unit shall not perform another background pre-scan operation unless the EN_PS bit is set to zero, then set to one, and another power on occurs.

The BACKGROUND MEDIUM SCAN INTERVAL TIME field specifies the minimum time, in hours, between the start of one background scan operation and the start of the next background medium scan operation. If the current background scan operation takes longer than the value specified in the BACKGROUND MEDIUM SCAN INTERVAL TIME field, then the logical unit shall:

a) continue the current background scan operation until that background scan operation is complete; and
b) start the next background medium scan operation upon completion of the current background scan operation.

The BACKGROUND PRE-SCAN TIME LIMIT field specifies the maximum time, in hours, for a background pre-scan operation to complete. If the background pre-scan operation does not complete within the specified time then the device server shall halt the background pre-scan operation. A value of zero specifies an unlimited timeout value.

The MINIMUM IDLE TIME BEFORE BACKGROUND SCAN field specifies the time, in milliseconds, that the logical unit shall be idle after suspending a background scan operation before resuming a background scan operation (e.g., after the device server has completed all of the commands in all task sets).

The MAXIMUM TIME TO SUSPEND BACKGROUND SCAN field specifies the time, in milliseconds, that the device server should take to start processing a command received while a logical unit is performing a background scan operation.

### 6.5.5 Caching mode page

The Caching mode page (see table 187) defines the parameters that affect the use of the cache.

**Table 187 — Caching mode page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS | SPF (0b) | PAGE CODE (08h) | | | | | |
| 1 | PAGE LENGTH (12h) | | | | | | | |
| 2 | IC | ABPF | CAP | DISC | SIZE | WCE | MF | RCD |
| 3 | DEMAND READ RETENTION PRIORITY | | | | WRITE RETENTION PRIORITY | | | |
| 4 | (MSB) | DISABLE PRE-FETCH TRANSFER LENGTH | | | | | | |
| 5 | | | | | | | | (LSB) |
| 6 | (MSB) | MINIMUM PRE-FETCH | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | (MSB) | MAXIMUM PRE-FETCH | | | | | | |
| 9 | | | | | | | | (LSB) |
| 10 | (MSB) | MAXIMUM PRE-FETCH CEILING | | | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | FSW | LBCSS | DRA | Vendor specific | | SYNC_PROG | | NV_DIS |
| 13 | NUMBER OF CACHE SEGMENTS | | | | | | | |
| 14 | (MSB) | CACHE SEGMENT SIZE | | | | | | |
| 15 | | | | | | | | (LSB) |
| 16 | Reserved | | | | | | | |
| 17 | Obsolete | | | | | | | |
| ••• | | | | | | | | |
| 19 | | | | | | | | |

The parameters saveable (PS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are defined in SPC-4.

The SPF bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field shall be set to the values shown in table 187.

An initiator control (IC) enable bit set to one specifies that the device server use one of the following fields to control the caching algorithm rather than the device server's own adaptive algorithm:

a) the NUMBER OF CACHE SEGMENTS field, if the SIZE bit is set to zero; or
b) the CACHE SEGMENT SIZE field, if the SIZE bit is set to one.

An abort pre-fetch (ABPF) bit set to one when the DRA bit is set to zero specifies that the device server abort a pre-fetch upon receipt of a new command. An ABPF bit set to one takes precedence over the value specified in the MINIMUM PRE-FETCH field. An ABPF bit set to zero when the DRA bit set to zero specifies that the termination of any active pre-fetch is dependent upon Caching mode page bytes 4 through 11 and is vendor specific.

A caching analysis permitted (CAP) bit set to one specifies that the device server perform caching analysis during subsequent operations. A CAP bit set to zero specifies that caching analysis be disabled (e.g., to reduce overhead time or to prevent nonpertinent operations from impacting tuning values).

A discontinuity (DISC) bit set to one specifies that the device server continue the pre-fetch across time discontinuities (e.g., across cylinders) up to the limits of the buffer, or segment, space available for the pre-fetch. A DISC bit set to zero specifies that pre-fetches be truncated or wrapped at time discontinuities.

A size enable (SIZE) bit set to one specifies that the CACHE SEGMENT SIZE field be used to control caching segmentation. A SIZE bit set to zero specifies that the NUMBER OF CACHE SEGMENTS field be used to control caching segmentation. Simultaneous use of both the number of segments and the segment size is vendor specific.

A writeback cache enable (WCE) bit set to zero specifies that the device server shall complete a ~~WRITE~~write command with GOOD status only performing all write medium operations for the command without error. A WCE bit set to one specifies that the device server may complete a write command with GOOD status after transferring the data for the command from the Data-Out Buffer to cache without error and prior to performing all write medium operations for the command. If an application client changes the WCE bit from one to zero via a MODE SELECT command, then the device server shall perform write medium operations on any logical block data in cache that is not the same as the logical block data referenced by the corresponding LBAs on the medium before completing the command.

A multiplication factor (MF) bit set to zero specifies that the device server shall interpret the MINIMUM PRE-FETCH field and the MAXIMUM PRE-FETCH field in terms of the number of logical blocks for each of the respective types of pre-fetch. An MF bit set to one specifies that the device server shall interpret the MINIMUM PRE-FETCH field and the MAXIMUM PRE-FETCH field to be specified in terms of a scalar number that, when multiplied by the number of logical blocks to be transferred for the current command, yields the number of logical blocks for each of the respective types of pre-fetch.

A read cache disable (RCD) bit set to zero specifies that the device server may use data requested by a read command (e.g., for transfer to the Data-In Buffer) from either cache or the medium. An RCD bit set to one specifies that the device server shall perform read medium operations for all of the data requested by a read command.

The DEMAND READ RETENTION PRIORITY field (see table 188) specifies the retention priority the device server should assign for data read into the cache that has also been transferred to the Data-In Buffer.

**Table 188 — DEMAND READ RETENTION PRIORITY field**

| Code | Description |
|---|---|
| 0h | The device server should not distinguish between retaining the indicated data and data placed into the cache by other means (e.g., pre-fetch). |
| 1h | The device server should replace data put into the cache via a READ command sooner (i.e., read data has lower priority) than data placed into the cache by other means (e.g., pre-fetch). |
| 2h to Eh | Reserved |
| Fh | The device server should not replace data put into the cache via a READ command if there is other data in the cache that was placed into the cache by other means (e.g., pre-fetch) and the data in the cache may be replaced. |

The WRITE RETENTION PRIORITY field (see table 189) specifies the retention priority the device server should assign for data written into the cache that has also been transferred from the cache to the medium.

**Table 189 — WRITE RETENTION PRIORITY field**

| Code | Description |
|---|---|
| 0h | The device server should not distinguish between retaining the indicated data and data placed into the cache by other means (e.g., pre-fetch). |
| 1h | The device server should replace data put into the cache during a WRITE command or a WRITE AND VERIFY command sooner (i.e., has lower priority) than data placed into the cache by other means (e.g., pre-fetch). |
| 2h to Eh | Reserved |
| Fh | The device server should not replace data put into the cache during a WRITE command or a WRITE AND VERIFY command if there is other data in the cache that was placed into the cache by other means (e.g., pre-fetch) and the data in the cache may be replaced. |

An anticipatory pre-fetch occurs when data is placed in the cache that has not been requested. This may happen in conjunction with the reading of data that has been requested. The DISABLE PRE-FETCH TRANSFER LENGTH field, the MINIMUM PRE-FETCH field, the MAXIMUM PRE-FETCH field, and the MAXIMUM PRE-FETCH CEILING field give an indication to the device server how it should manage the cache based on the most recent READ command. An anticipatory pre-fetch may occur based on other information. These fields are only recommendations to the device server and should not cause a CHECK CONDITION to occur if the device server is not able to satisfy the request.

The DISABLE PRE-FETCH TRANSFER LENGTH field specifies the selective disabling of anticipatory pre-fetch on long transfer lengths. The value in this field is compared to the transfer length requested by a READ command. If the transfer length is greater than the disable pre-fetch transfer length, then an anticipatory pre-fetch is not done for the command. Otherwise the device server should attempt an anticipatory pre-fetch. If the DISABLE PRE-FETCH TRANSFER LENGTH field is set to zero, then all anticipatory pre-fetching is disabled for any request for data, including those with a transfer length of zero.

The MINIMUM PRE-FETCH field specifies the number of logical blocks to pre-fetch regardless of the delays it might cause in processing subsequent commands. The field contains either:

a) a number of logical blocks, if the MF bit is set to zero; or
b) a scalar multiplier of the value in the TRANSFER LENGTH field, if the MF bit is set to one.

The pre-fetching operation begins at the logical block after the last logical block of a READ command. Pre-fetching shall always halt when it reaches the last logical block on the medium. Errors that occur during the pre-fetching operation shall not be reported to the application client unless the device server is unable to process subsequent commands correctly as a result of the error. In this case the error may be reported either as an error for that subsequent command, or as a deferred error, at the discretion of the device server and according to the rules for reporting deferred errors (see SPC-4).

If the pre-fetch has read more than the amount of data specified by the MINIMUM PRE-FETCH field, then pre-fetching should be terminated whenever another command enters the enabled state (see SAM-5). This requirement is ignored when the MINIMUM PRE-FETCH field value is equal to the MAXIMUM PRE-FETCH field value.

The MAXIMUM PRE-FETCH field specifies the number of logical blocks to pre-fetch if the pre-fetch does not delay processing of subsequent commands. The field contains either:

a) a number of logical blocks, if the MF bit is set to zero; or
b) a scalar multiplier of the value in the TRANSFER LENGTH field, if the MF bit is set to one.

The MAXIMUM PRE-FETCH field contains the maximum amount of data to pre-fetch as a result of one READ command. It is used in conjunction with the DISABLE PRE-FETCH TRANSFER LENGTH field and MAXIMUM PRE-FETCH CEILING field to trade off pre-fetching new data with displacing old data already stored in the cache.

The MAXIMUM PRE-FETCH CEILING field specifies an upper limit on the number of logical blocks computed as the maximum pre-fetch. If this number of logical blocks is greater than the value in the MAXIMUM PRE-FETCH field, then the number of logical blocks to pre-fetch shall be truncated to the value stored in the MAXIMUM PRE-FETCH CEILING field.

NOTE 32 - If the MF bit is set to one, then the MAXIMUM PRE-FETCH CEILING field is useful in limiting the amount of data to be pre-fetched.

A force sequential write (FSW) bit set to one specifies that, for commands writing to more than one logical block, the device server shall write the logical blocks to the medium in ascending sequential order. An FSW bit set to zero specifies that the device server may reorder the sequence of writing logical blocks (e.g., in order to achieve faster command completion).

A logical block cache segment size (LBCSS) bit set to one specifies that the CACHE SEGMENT SIZE field units shall be interpreted as logical blocks. An LBCSS bit set to zero specifies that the CACHE SEGMENT SIZE field units shall be interpreted as bytes. The LBCSS shall not impact the units of other fields.

A disable read-ahead (DRA) bit set to one specifies that the device server shall not read into the pre-fetch buffer any logical blocks beyond the addressed logical block(s). A DRA bit set to zero specifies that the device server may continue to read logical blocks into the pre-fetch buffer beyond the addressed logical block(s).

The synchronize cache progress indication support (SYNC_PROG) field (see table 190) specifies device server progress indication reporting while a SYNCHRONIZE CACHE command (see 5.26 and 5.27) is being processed.

**Table 190 — SYNC_PROG field**

| Code | Description |
|---|---|
| 00b | The device server shall not terminate commands due to the synchronize cache operation and shall not provide pollable sense data. |
| 01b | The device server:<br>a) shall not terminate commands due to the synchronize cache operation; and<br>b) shall provide pollable sense data with the sense key set to NO SENSE, the additional sense code set to SYNCHRONIZE CACHE OPERATION IN PROGRESS, and the PROGRESS INDICATION field set to indicate the progress of the synchronize cache operation. |
| 10b | The device server:<br>a) shall process INQUIRY commands, REPORT LUNS commands, REPORT TARGET PORT GROUPS commands, and REQUEST SENSE commands;<br>b) may process commands that do not require resources used for the synchronize cache operation;<br>c) shall terminate commands that require resources used for the synchronize cache operation with CHECK CONDITION status with the sense key set to NOT READY, the additional sense code set to LOGICAL UNIT NOT READY, SYNCHRONIZE CACHE OPERATION IN PROGRESS, and the PROGRESS INDICATION field set to indicate the progress of the synchronize cache operation; and<br>d) shall provide pollable sense data with the sense key set to NOT READY, the additional sense code set to LOGICAL UNIT NOT READY, SYNCHRONIZE CACHE OPERATION IN PROGRESS, and the PROGRESS INDICATION field set to indicate the progress of the synchronize cache operation. |
| 11b | Reserved |

An NV_DIS bit set to one specifies that the device server shall disable a non-volatile cache and indicates that a non-volatile cache is supported but disabled. An NV_DIS bit set to zero specifies that the device server may use a non-volatile cache and indicates that a non-volatile cache may be present and enabled.

The NUMBER OF CACHE SEGMENTS field specifies the number of segments into which the device server shall divide the cache.

The CACHE SEGMENT SIZE field specifies the segment size in bytes if the LBCSS bit is set to zero or in logical blocks if the LBCSS bit is set to one. The CACHE SEGMENT SIZE field is valid only when the SIZE bit is set to one.

### 6.5.6 Informational Exceptions Control mode page

The Informational Exceptions Control mode page (see table 191) defines the methods used by the device server to control the processing and reporting of informational exception conditions. Informational exception conditions are defined as any event that the device server reports or logs as failure predictions (i.e., with the ADDITIONAL SENSE CODE field set to 5Dh (e.g., FAILURE PREDICTION THRESHOLD EXCEEDED)) or warnings (i.e., with the ADDITIONAL SENSE CODE field set to 0Bh (e.g., WARNING)).

Informational exception conditions may occur while a logical unit is processing:

a) a background self-test (see SPC-4);
b) device specific background functions (see SPC-4); or
c) other device specific events.

An informational exception condition may occur at any time (e.g., the condition may be asynchronous to any commands issued by an application client).

The mode page policy for this mode page shall be shared or per I_T nexus (see SPC-4).

NOTE 33 - Storage devices that support SMART (Self-Monitoring Analysis and Reporting Technology) for predictive failure software should use informational exception conditions.

**Table 191 — Informational Exceptions Control mode page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS | SPF (0b) | PAGE CODE (1Ch) | | | | | |
| 1 | PAGE LENGTH (0Ah) | | | | | | | |
| 2 | PERF | Reserved | EBF | EWASC | DEXCPT | TEST | EBACKERR | LOGERR |
| 3 | Reserved | | | | MRIE | | | |
| 4 | (MSB) | INTERVAL TIMER | | | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | (MSB) | REPORT COUNT | | | | | | |
| ••• | | | | | | | | |
| 11 | | | | | | | | (LSB) |

The PS bit, the SPF bit, the PAGE CODE field, and the PAGE LENGTH field are described in SPC-4.

The SPF bit, the PAGE CODE field, and the PAGE LENGTH field shall be set to the values shown in table 191.

If the performance (PERF) bit is set to zero, then the device server may process informational exception conditions that cause delays in processing other operations (e.g., processing a command). If the PERF bit is set to one, then the device server shall not process informational exception conditions that cause delays in processing other operations. A PERF bit set to one may cause the device server to disable some or all of the processing of informational exception conditions, thereby limiting the reporting of informational exception conditions.

If device specific background functions (see SPC-4) are implemented by the logical unit, and the enable background function (EBF) bit is set to one, then the device server shall enable device specific background functions. If the EBF bit is set to zero, then the device server shall disable device specific background functions. Background functions with separate enable control bits (e.g., the background medium scan (see 4.24)) are not controlled by the EBF bit.

The enable warning (EWASC) bit specifies if the device server enables reporting of warnings (see table 192).

The disable exception control (DEXCPT) bit specifies if the device server enables reporting of failure predictions (see table 192).

The TEST bit specifies if the device server creates a test device failure prediction (see table 192).

If an informational exception condition occurs that is not the result of the logical unit processing a background self-test (see SPC-4) or device specific background function (see SPC-4), then the device server:

a) shall use the definitions for the combination of the values in the EWASC bit, the DEXCPT bit, and the TEST bit shown in table 192 for processing informational exception conditions when the MRIE field is set from 2h to 6h;
b) may use the definitions for the combination of the values in the EWASC bit, the DEXCPT bit, and the TEST bit shown table 192 for processing informational exception conditions when the MRIE field is set from Ch to Fh; and
c) shall ignore the EWASC bit, the DEXCPT bit, and the TEST bit when the MRIE field is set to any value other than 2h to 6h, or Ch to Fh.

**Table 192 — Definitions for the combinations of values in EWASC, DEXCPT, and TEST**

| EWASC | DEXCPT | TEST | Description |
|---|---|---|---|
| 0 | 0 | 0 | The device server shall process informational exception conditions as follows:<br>a) failure prediction processing shall be enabled [a]; and<br>b) warning processing shall be disabled. |
| 1 | 0 | 0 | The device server shall process informational exception conditions as follows:<br>a) failure prediction processing shall be enabled [a]; and<br>b) warning processing shall be enabled [a]. |
| 0 | 1 | 0 | The device server shall process informational exception conditions as follows:<br>a) failure prediction processing shall be disabled; and<br>b) warning processing shall be disabled. |
| 1 | 1 | 0 | The device server shall process informational exception conditions as follows:<br>a) failure prediction processing shall be disabled; and<br>b) warning processing shall be enabled [a]. |
| 0 | 0 | 1 | The device server shall set the additional sense code to FAILURE PREDICTION THRESHOLD EXCEEDED (FALSE) [a]. |
| 1 | 0 | 1 | |
| 0 | 1 | 1 | The device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST. |
| 1 | 1 | 1 | |

[a] If applicable based on the value in the MRIE field (e.g., 2h to 6h), then the values in the LOGERR bit, the INTERVAL TIMER field, and the REPORT COUNT field determine how the informational exception condition is processed.

If an informational exception condition occurs while the logical unit is processing a background self-test (see SPC-4) or background function (see SPC-4), then the enable background error (EBACKERR) bit determines how the device server processes the informational exception as defined in the following:

a) if the EBACKERR bit is set to zero, then the device server shall disable reporting of informational exception conditions that occur during the processing of background self-tests and background functions;
b) if the EBACKERR bit is set to one, then, for informational exception conditions that occur during the processing of background self-tests and background functions, the device server shall:
   A) enable reporting of the informational exception conditions;
   B) use the method for reporting the informational exception conditions as determined by contents of the MRIE field; and

C) report the informational exception conditions as soon as the method specified in the MRIE field occurs (i.e., the INTERVAL TIMER field and REPORT COUNT field do not apply for background self-test errors and errors that occur during background functions);

and

c) logging by the device server of informational exception conditions is determined by the value in the LOGERR bit.

The log errors (LOGERR) bit determines how a device server logs informational exception conditions when the MRIE field is set to any value supported by the device server. If the LOGERR bit is set to zero, then the device server may log any informational exception conditions in the Informational Exceptions log page (see SPC-4). If the LOGERR bit is set to one, then the device server shall log informational exception conditions in the Informational Exceptions log page.

The method of reporting informational exceptions (MRIE) field (see table 193) defines the method that shall be used by the device server to report:

a) informational exception conditions when the specified code value is supported by the device server; and
b) background self-test errors and device specific background function errors with the ADDITIONAL SENSE CODE field set to 0Bh or 5Dh when the EBACKERR bit is set to one, and the specified code value is supported by the device server.

A device server that supports the Informational Exceptions Control mode page shall support at least one code value in the MRIE field.

The priority of reporting multiple informational exceptions is vendor specific.

**Table 193 — Method of reporting informational exceptions (MRIE) field** (part 1 of 2)

| Code | Description |
|---|---|
| 0h | **No reporting of informational exception condition:** The device server shall not report information exception conditions. |
| 1h | Obsolete |
| 2h | **Establish unit attention condition:** The device server shall report informational exception conditions by establishing a unit attention condition (see SAM-5) for the initiator port associated with every I_T nexus, with the additional sense code set to indicate the cause of the informational exception condition.[a] |
| 3h | **Conditionally generate recovered error:** The device server shall report informational exception conditions, if the reporting of recovered errors is allowed [b], by modifying the completion of the next command processed without encountering any errors, regardless of the I_T nexus on which the command was received. The modification shall be to terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR and the additional sense code set to indicate the cause of the informational exception condition. |
| 4h | **Unconditionally generate recovered error:** The device server shall report informational exception conditions, regardless of whether the reporting of recovered errors is allowed [b], by modifying the completion of the next command processed without encountering any errors, regardless of the I_T nexus on which the command was received. The modification shall be to terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR and the additional sense code set to indicate the cause of the informational exception condition. |

[a] The device server terminates the command to report the unit attention condition for the informational exception condition (i.e., the device server does not process the command except to report the unit attention condition) (see SAM-5).

[b] This is controlled by the PER bit in the Read-Write Error Recovery mode page (see 6.5.8).

**Table 193 — Method of reporting informational exceptions (MRIE) field** (part 2 of 2)

| Code | Description |
|---|---|
| 5h | **Generate no sense:** The device server shall report informational exception conditions by modifying the completion of the next command processed without encountering any errors, regardless of the I_T nexus on which the command was received. The modification shall be to terminate the command with CHECK CONDITION status with the sense key set to NO SENSE and the additional sense code set to indicate the cause of the informational exception condition. |
| 6h | **Only report informational exception condition on request:** The device server shall provide informational exception(s) information with the sense key set to NO SENSE and the additional sense code set to indicate the cause of the informational exception condition for return in response to a REQUEST SENSE command as described in SPC-4. To find out about information exception conditions, the application client polls the device server by issuing a REQUEST SENSE command. |
| 7h to Bh | Reserved |
| Ch to Fh | Vendor specific |

[a] The device server terminates the command to report the unit attention condition for the informational exception condition (i.e., the device server does not process the command except to report the unit attention condition) (see SAM-5).
[b] This is controlled by the PER bit in the Read-Write Error Recovery mode page (see 6.5.8).

The INTERVAL TIMER field specifies the period in 100 millisecond increments that the device server shall use for reporting that an informational exception condition has occurred (see table 194). After an informational exception condition has been reported, the interval timer shall be started. An INTERVAL TIMER field set to zero or FFFF_FFFFh specifies that the period for reporting an informational exception condition is vendor specific.

The REPORT COUNT field specifies the maximum number of times the device server may report an informational exception condition to the application client. A REPORT COUNT field set to zero specifies that there is no limit on the number of times the device server may report an informational exception condition.

The device server shall use the values in the INTERVAL TIMER field and the REPORT COUNT field based on the value in the MRIE field as shown in table 194.

**Table 194 — Use of the INTERVAL TIMER field and the REPORT COUNT field based on the MRIE field**

| MRIE [a] | Description |
|---|---|
| 2h to 6h | If processing of an informational exception condition is enabled (see table 193), then the device server shall:<br>1) report an informational exception condition when the condition is first detected; and<br>2) if the value in the REPORT COUNT field is not equal to one, then:<br>  1) if the interval timer field is not set to zero or FFFF_FFFFh, then wait the time specified in the INTERVAL TIMER field, and, if that informational exception condition still exists, report the informational exception again; and<br>  2) while the informational exception condition exists, continue to report the informational exception condition after waiting the time specified in the INTERVAL TIMER field until the condition has been reported the number of times specified by the REPORT COUNT field. |
| Ch to Fh | The device server may use or may ignore the values in the INTERVAL TIMER field and the REPORT COUNT field to report the informational exception condition based on the device specific implementation. |

[a] For values in the MRIE field (see table 193) not shown in this table, the INTERVAL TIMER field and the REPORT COUNT field shall be ignored.

Maintaining the interval timer and the report counter across power cycles, hard resets, logical unit resets, and I_T nexus losses by the device server is vendor specific.

### 6.5.7 Logical Block Provisioning mode page

#### 6.5.7.1 Introduction

The Logical Block Provisioning mode page (see table 195) specifies the parameters that a device server that supports logical block provisioning threshold values (see 4.7.3.8) shall use to report logical block provisioning threshold notifications (see 4.7.3.8.4). The mode page policy (see SPC-4) for this page shall be shared.

If a method not defined by this standard changes the parameter data to be returned by a device server in the Logical Block Provisioning mode page, then the device server shall establish a unit attention condition for the initiator port associated with every I_T nexus with the additional sense code set to MODE PARAMETERS CHANGED.

**Table 195 — Logical Block Provisioning mode page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS | SPF (1b) | PAGE CODE (1Ch) | | | | | |
| 1 | SUBPAGE CODE (02h) | | | | | | | |
| 2 | (MSB) | | PAGE LENGTH (n - 3) | | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | Reserved | | | | | | | SITUA |
| 5 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | |
| **Threshold descriptors** | | | | | | | | |
| 16 | Threshold descriptor [first] (see 6.5.7.2) | | | | | | | |
| ••• | | | | | | | | |
| 23 | | | | | | | | |
| | ⋮ | | | | | | | |
| n - 7 | Threshold descriptor [last] (see 6.5.7.2) | | | | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | |

The parameters saveable (PS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are defined in SPC-4.

The SPF bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field shall be set to the values shown in table 195.

A single initiator threshold unit attention (SITUA) bit set to one indicates that the logical block provisioning threshold notification unit attention condition is established on a single I_T nexus as described in 4.7.3.8.4. A SITUA bit set to zero indicates that the logical block provisioning threshold notification unit attention condition is established on multiple I_T nexuses as described in 4.7.3.8.4.

The threshold descriptors are defined in 6.5.7.2.

### 6.5.7.2 Threshold descriptor format

The threshold descriptor format is described in table 196.

**Table 196 — Threshold descriptor format**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | ENABLED | Reserved | THRESHOLD TYPE | | | THRESHOLD ARMING | | |
| 1 | THRESHOLD RESOURCE | | | | | | | |
| 2 | Reserved | | | | | | | |
| 3 | | | | | | | | |
| 4 | (MSB) | | | | | | | |
| ••• | THRESHOLD COUNT | | | | | | | |
| 7 | | | | | | | | (LSB) |

An ENABLED bit set to one specifies that the threshold is enabled. An ENABLED bit set to zero specifies that the threshold is disabled.

The THRESHOLD TYPE field (see table 197) specifies the type of this threshold.

**Table 197 — THRESHOLD TYPE field**

| Code | Description |
|---|---|
| 000b | The THRESHOLD COUNT field specifies a soft threshold and, if the threshold is enabled, then, when the threshold is reached, the device server shall establish a unit attention condition as described in 4.7.3.8.4. |
| All others | Reserved |

The THRESHOLD ARMING field (see table 198) specifies the arming method used for operation of this threshold.

**Table 198 — THRESHOLD ARMING field**

| Code | Description | Reference |
|---|---|---|
| 000b | The threshold operates as an armed decreasing threshold. | 4.7.3.8.2 |
| 001b | The threshold operates as an armed increasing threshold. | 4.7.3.8.3 |
| All others | Reserved | |

The THRESHOLD RESOURCE field specifies the resource of this threshold. The contents of this field are as defined for parameters codes less than 0100h (see table 163) in the Logical Block Provisioning log page (see 6.4.4).

The valid combinations of the THRESHOLD TYPE field, the THRESHOLD ARMING field, and the THRESHOLD RESOURCE field are shown in table 4 (see 4.7.3.8.1).

The THRESHOLD COUNT field specifies the center of the threshold range for this threshold expressed as a number of threshold sets (i.e., the number LBA mapping resources expressed as a number of threshold sets).

### 6.5.8 Read-Write Error Recovery mode page

The Read-Write Error Recovery mode page (see table 199) specifies the error recovery parameters the device server shall use during:

a) read medium operations that are not verify medium operations; or
b) write medium operations (e.g., from WRITE commands or the write operation of a WRITE AND VERIFY command).

**Table 199 — Read-Write Error Recovery mode page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS | SPF (0b) | PAGE CODE (01h) | | | | | |
| 1 | PAGE LENGTH (0Ah) | | | | | | | |
| 2 | AWRE | ARRE | TB | RC | **Error recovery bits** | | | |
| | | | | | EER | PER | DTE | DCR |
| 3 | READ RETRY COUNT | | | | | | | |
| 4 | Obsolete | | | | | | | |
| 5 | Obsolete | | | | | | | |
| 6 | Obsolete | | | | | | | |
| 7 | LBPERE | Reserved | | | | | Restricted for MMC-6 | |
| 8 | WRITE RETRY COUNT | | | | | | | |
| 9 | Reserved | | | | | | | |
| 10 | (MSB) | RECOVERY TIME LIMIT | | | | | | |
| 11 | | | | | | | | (LSB) |

The parameters saveable (PS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are defined in SPC-4.

The SPF bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field shall be set to the values shown in table 199.

An automatic write reassignment enabled (AWRE) bit set to zero specifies that the device server shall not perform automatic write reassignment.

An AWRE bit set to one specifies that the device server shall enable automatic write reassignment for LBAs referencing logical blocks for which a recovered error or unrecovered error occurs during a write medium operation. Automatic write reassignment shall be performed only if the device server has the valid data (e.g., original data in a buffer or recovered from the medium). The valid data shall be placed in the logical block referenced by the reassigned LBA. The device server shall report any failures that occur during the reassignment operation. Error reporting as specified by the error recovery bits (i.e., the EER bit, the PER bit, the DTE bit, and the DCR bit) shall be performed only after completion of the reassign operation.

An automatic read reassignment enabled (ARRE) bit set to zero specifies that the device server shall not perform automatic read reassignment.

An ARRE bit set to one specifies that the device server shall enable automatic read reassignment for LBAs referencing logical blocks for which a recovered error occurs during a read medium operation. All error recovery actions required by the error recovery bits shall be processed. Automatic read reassignment shall then be performed only if the device server successfully recovers the data. The recovered data shall be placed in the logical block referenced by the reassigned LBA. The device server shall report any failures that occur during the reassign operation. Error reporting as specified by the error recovery bits shall be performed only after completion of the reassign operation.

A transfer block (TB) bit set to zero specifies that if an unrecovered read error occurs during a read medium operation, then the device server shall not transfer any data for the logical block to the Data-In Buffer. A TB bit set to one specifies that if an unrecovered read error occurs during a read medium operation, then the device server shall transfer pseudo read data before returning CHECK CONDITION status. The data returned in this case is vendor specific. The TB bit does not affect the action taken for recovered read errors.

A read continuous (RC) bit set to zero specifies that error recovery operations that cause delays during the data transfer are acceptable. Data shall not be fabricated.

An RC bit set to one specifies the device server shall transfer the entire requested length of data without adding delays during the data transfer to perform error recovery procedures. The device server may transfer pseudo read data in order to maintain a continuous flow of data. The device server shall assign priority to the RC bit over conflicting bits within this byte.

NOTE 34 - The RC bit when set to one is useful for image processing, audio, or video applications.

An enable early recovery (EER) bit set to one specifies that the device server shall use the most expedient form of error recovery first. An EER bit set to zero specifies that the device server shall use an error recovery procedure that minimizes the risk of error mis-detection or mis-correction. This bit only applies to data error recovery and it does not affect positioning retries.

NOTE 35 - An EER bit set to one allows an increase in the probability of error mis-detection or mis-correction. An EER bit set to zero allows the specified retry limit to be exhausted prior to using additional information (e.g., ECC bytes) to correct the error.

A post error (PER) bit set to one specifies that if a recovered read error occurs while processing a read command or a write command, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR. A PER bit set to zero specifies that if a recovered read error occurs while processing a read command or a write command, then the device server shall perform error recovery procedures within the limits established by the error recovery parameters and not terminate the command that specified that operation with CHECK CONDITION status with the sense key set to RECOVERED ERROR (e.g., the device server may terminate the command that specified the operation with CHECK CONDITION status with the sense key set to MEDIUM ERROR if the error becomes uncorrectable based on the established limits during the error recovery process). If the DTE bit is set to one, then the PER bit shall be set to one.

A data terminate on error (DTE) bit set to one specifies that, upon detection of a recovered error, the device server shall terminate the data transfer to the Data-In Buffer for a read command or the data transfer to the Data-Out Buffer for a write command upon detection of a recovered error. A DTE bit set to zero specifies that, upon detection of a recovered error, the device server shall not terminate the data transfer to the Data-In Buffer for a read command or the data transfer to the Data-Out Buffer for a write command.

A disable correction (DCR) bit set to one specifies that additional information (e.g., ECC bytes) (see 4.5) shall not be used for data error recovery. A DCR bit set to zero allows the use of additional information (e.g., ECC bytes) for data error recovery. If the EER bit is set to one, the DCR bit shall be set to zero.

The combinations of the error recovery bits (i.e., the EER bit, the PER bit, the DTE bit, and the DCR bit) are defined in table 200.

**Table 200 — Error recovery bit combinations** (part 1 of 4)

| EER | PER | DTE | DCR | Description |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read medium operations, the WRITE RETRY COUNT field for write medium operations, and the VERIFY RETRY COUNT field (see 6.5.9) for verify medium operations and shall perform error correction in an attempt to recover the data.<br><br>The device server shall not report recovered read or write errors. The device server shall terminate a command performing a read medium operation or a write medium operation with CHECK CONDITION status before the transfer count is exhausted only if an unrecovered error is detected.<br><br>If an unrecovered read error occurs during a read medium operation, then the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the Data-In Buffer. |
| 0 | 0 | 0 | 1 | The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read medium operations, the WRITE RETRY COUNT field for write medium operations, and the VERIFY RETRY COUNT field (see 6.5.9) for verify medium operations but shall not perform error correction in an attempt to recover the data.<br><br>The device server shall not report recovered read errors or write errors. The device server shall terminate a command performing a read medium operation or write medium operation with CHECK CONDITION status before the transfer count is exhausted only if an unrecovered error is detected.<br><br>If an unrecovered read error occurs during a read medium operation, then the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the Data-In Buffer. |
| 0 | 0 | 1 | 0 | Invalid mode. The PER bit shall be set to one if the DTE bit is set to one. [a] |
| 0 | 0 | 1 | 1 | |

[a] If an invalid combination of the error recovery bits is sent by an application client, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

**Table 200 — Error recovery bit combinations** (part 2 of 4)

| EER | PER | DTE | DCR | Description |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read medium operations, the VERIFY RETRY COUNT field (see 6.5.9) for verify medium operations, the WRITE RETRY COUNT field for write medium operations, and shall perform error correction in an attempt to recover the data.<br><br>The device server shall terminate a command performing a read medium operation or write medium operation with CHECK CONDITION status before the transfer count is exhausted only if an unrecovered error is detected.<br><br>If an unrecovered read error occurs during a read medium operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the Data-In Buffer.<br><br>If a recovered error occurs while the device server is performing a read medium operation or write medium operation, then, after the operation is complete, the device server shall terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR. The INFORMATION field in the sense data shall contain the LBA of the last recovered error that occurred during the command. |
| 0 | 1 | 0 | 1 | The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read medium operations, the VERIFY RETRY COUNT field (see 6.5.9) for verify medium operations, and the WRITE RETRY COUNT field for write medium operations, but shall not perform error correction in an attempt to recover the data.<br><br>The device server shall terminate a command performing a read medium operation or write medium operation with CHECK CONDITION status before the transfer count is exhausted only if an unrecovered error is detected.<br><br>If an unrecovered read error occurs during a read medium operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the Data-In Buffer.<br><br>If a recovered error occurs while the device server is performing a read medium operation or write medium operation, then, after the operation is complete, the device server shall terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR. The INFORMATION field in the sense data shall contain the LBA of the last recovered error that occurred during the command. |

[a] If an invalid combination of the error recovery bits is sent by an application client, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

**Table 200 — Error recovery bit combinations** (part 3 of 4)

| EER | PER | DTE | DCR | Description |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read medium operations, the VERIFY RETRY COUNT field (see 6.5.9) for verify medium operations, the WRITE RETRY COUNT field for write medium operations, and shall perform error correction in an attempt to recover the data.<br><br>The device server shall terminate a command performing a read medium operation or write medium operation with CHECK CONDITION status before the transfer count is exhausted if any error, either recovered or unrecovered, is detected. The INFORMATION field in the sense data shall contain the LBA of error.<br><br>If an unrecovered read error occurs during a read medium operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the Data-In Buffer. |
| 0 | 1 | 1 | 1 | The device server shall perform the full number of retries as specified in the READ RETRY COUNT field for read medium operations, the VERIFY RETRY COUNT field (see 6.5.9) for verify medium operations, the WRITE RETRY COUNT field for write medium operations, but shall not perform error correction in an attempt to recover the data.<br><br>The device server shall terminate a command performing a read medium operation or write medium operation with CHECK CONDITION status before the transfer count is exhausted if any error, either recovered or unrecovered, is detected. The INFORMATION field in the sense data shall contain the LBA of the error.<br><br>If an unrecovered read error occurs during a read medium operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the Data-In Buffer. |
| 1 | 0 | 0 | 0 | The device server shall perform the fewest possible number of retries and perform error correction in an attempt to recover the data.<br><br>The device server shall not report recovered read or write errors. The device server shall terminate a command performing a read medium operation or write medium operation with CHECK CONDITION status before the transfer count is exhausted only if an unrecovered error is detected.<br><br>If an unrecovered read error occurs during a read medium operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the Data-In Buffer. |
| 1 | 0 | 0 | 1 | Invalid mode. The DCR bit shall be set to zero if the EER bit is set to one. [a] |
| 1 | 0 | 1 | 0 | Invalid mode. The PER bit shall be set to one if the DTE bit is set to one. [a] |
| 1 | 0 | 1 | 1 | |

[a] If an invalid combination of the error recovery bits is sent by an application client, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

**Table 200 — Error recovery bit combinations** (part 4 of 4)

| EER | PER | DTE | DCR | Description |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | The device server shall perform the fewest possible number of retries and perform error correction in an attempt to recover the data.<br><br>The device server shall terminate a command performing a read medium operation or write medium operation with CHECK CONDITION status before the transfer count is exhausted only if an unrecovered error is detected.<br><br>If an unrecovered read error occurs during a read medium operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the Data-In Buffer.<br><br>If a recovered error occurs while the device server is performing a read medium operation or write medium operation, then, after the operation is complete, the device server shall terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR. The INFORMATION field in the sense data shall contain the LBA of the last recovered error that occurred during the command. |
| 1 | 1 | 0 | 1 | Invalid mode. The DCR bit shall be set to zero if the EER bit is set to one. [a] |
| 1 | 1 | 1 | 0 | The device server shall perform the fewest possible number of retries and perform error correction in an attempt to recover the data.<br><br>The device server shall terminate a command performing a read medium operation or write medium operation with CHECK CONDITION status before the transfer count is exhausted if any error, either recovered or unrecovered, is detected. The INFORMATION field in the sense data shall contain the LBA of the error.<br><br>If an unrecovered read error occurs during a read medium operation, the transfer block (TB) bit determines whether the data for the logical block with the unrecovered read error is transferred to the Data-In Buffer. |
| 1 | 1 | 1 | 1 | Invalid mode. The DCR bit shall be set to zero if the EER bit is set to one. [a] |

[a] If an invalid combination of the error recovery bits is sent by an application client, then the device server shall terminate the MODE SELECT command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The READ RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during read medium operations.

The WRITE RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during write medium operations.

A logical block provisioning error reporting enabled (LBPERE) bit set to one specifies that logical block provisioning threshold notification is enabled. A LBPERE bit set to zero specifies that logical provisioning threshold notification is disabled (see 4.7.3.8.4).

The RECOVERY TIME LIMIT field specifies in milliseconds the maximum time duration that the device server shall use for data error recovery procedures. The device server may round this value as described in SPC-4. The limit in this field specifies the maximum error recovery time allowed for any individual logical block. A RECOVERY TIME LIMIT field set to zero specifies that the device server shall use its default value.

When both a retry count and a recovery time limit are specified, the field that specifies the recovery action of least duration shall have priority.

NOTE 36 - To disable all types of correction and retries, it is recommended that the application client set the EER bit to zero, the PER bit to one, the DTE bit to one, the DCR bit to one, the READ RETRY COUNT field to 00h, the WRITE RETRY COUNT field to 00h, and the RECOVERY TIME LIMIT field to 0000h.

### 6.5.9 Verify Error Recovery mode page

The Verify Error Recovery mode page (see table 201) specifies the error recovery parameters the device server shall use during verify medium operation (e.g., from VERIFY commands and the verify medium operation of the WRITE AND VERIFY commands). Verify medium operations do not trigger automatic read reassignment.

**Table 201 — Verify Error Recovery mode page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PS | SPF (0b) | PAGE CODE (07h) | | | | | |
| 1 | PAGE LENGTH (0Ah) | | | | | | | |
| 2 | Reserved | | | | Error recovery bits | | | |
| | | | | | EER | PER | DTE | DCR |
| 3 | VERIFY RETRY COUNT | | | | | | | |
| 4 | Obsolete | | | | | | | |
| 5 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 9 | | | | | | | | |
| 10 | (MSB) | VERIFY RECOVERY TIME LIMIT | | | | | | |
| 11 | | | | | | | | (LSB) |

The parameters saveable (PS) bit, the subpage format (SPF) bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field are defined in SPC-4.

The SPF bit, the PAGE CODE field, the SUBPAGE CODE field, and the PAGE LENGTH field shall be set to the values shown in table 201.

The EER bit, the PER bit, the DTE bit, and the DCR bit (i.e., the error recovery bits) are defined in 6.5.8. The combinations of these bits are defined in table 200 (see 6.5.8).

The VERIFY RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during a verify medium operation.

The VERIFY RECOVERY TIME LIMIT field specifies in milliseconds the maximum time duration that the device server shall use error recovery procedures to recover data for an individual logical block. The device server may round this value as described in SPC-4.

When both a retry count and a recovery time limit are specified, the one that requires the least time for data error recovery actions shall have priority.

NOTE 37 - To disable all types of correction and retries it is recommended that the application client set the EER bit to zero, the PER bit to one, the DTE bit to one, the DCR bit to one, the VERIFY RETRY COUNT field to 00h, and the VERIFY RECOVERY TIME LIMIT field to 0000h.

## 6.6 Vital product data (VPD) parameters

### 6.6.1 VPD parameters overview

This subclause defines the VPD pages used only with direct-access block devices. See SPC-4 for VPD pages used with all device types. The VPD pages and their corresponding page codes specific to direct-access block devices are defined in table 202.

**Table 202 — VPD page codes for direct-access block devices**

| VPD page name | Page code [a] | Reference | Support requirements |
|---|---|---|---|
| ATA Information | 89h | SAT-3 | See SAT-3 |
| Block Device Characteristics VPD page | B1h | 6.6.2 | Optional |
| Block Limits VPD page | B0h | 6.6.3 | Optional |
| Logical Block Provisioning VPD page | B2h | 6.6.4 | Optional |
| Power Consumption VPD page | See SPC-4 | SPC-4 | See SPC-4 |
| Referrals VPD page | B3h | 6.6.5 | Optional |
| Third-Party Copy VPD page | See SPC-4 | SPC-4 and 6.6.6 | Optional [b] |
| Reserved for this standard | B4h to BFh | | |

[a] All page code and subpage code combinations for direct-access block devices not shown in this table are reserved.
[b] Mandatory if the POPULATE TOKEN command and the WRITE USING TOKEN command are supported.

### 6.6.2 Block Device Characteristics VPD page

The Block Device Characteristics VPD page (see table 203) contains parameters indicating characteristics of the logical unit.

**Table 203 — Block Device Characteristics VPD page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PERIPHERAL QUALIFIER | | | PERIPHERAL DEVICE TYPE | | | | |
| 1 | PAGE CODE (B1h) | | | | | | | |
| 2 | (MSB) | | | PAGE LENGTH (003Ch) | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | (MSB) | | | MEDIUM ROTATION RATE | | | | |
| 5 | | | | | | | | (LSB) |
| 6 | PRODUCT TYPE | | | | | | | |
| 7 | WABEREQ | | WACEREQ | | NOMINAL FORM FACTOR | | | |
| 8 | Reserved | | | | | | | VBULS |
| 9 | Reserved | | | | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in SPC-4.

The PAGE CODE field and PAGE LENGTH field are defined in SPC-4 and shall be set to the values shown in table 203.

The MEDIUM ROTATION RATE field is defined in table 204.

**Table 204 — MEDIUM ROTATION RATE field**

| Code | Description |
|---|---|
| 0000h | Medium rotation rate is not reported |
| 0001h | Non-rotating medium (e.g., solid state) |
| 0002h to 0400h | Reserved |
| 0401h to FFFEh | Nominal medium rotation rate in rpm (e.g., 7 200 rpm = 1C20h,10 000 rpm = 2710h, and 15 000 rpm = 3A98h) |
| FFFFh | Reserved |

The PRODUCT TYPE field (see table 205) defines the product type of the storage device.

**Table 205 — PRODUCT TYPE field**

| Code | Description |
|---|---|
| 00h | Not specified |
| 01h to EFh | Reserved |
| F0h to FFh | Vendor specific |

If a write medium operation has not been performed on a logical block referenced by a mapped LBA since a sanitize block erase operation was performed, and no other error occurs during the processing of a read command specifying that LBA, then the write after block erase required (WABEREQ) field indicates the device server behavior (see table 206).

**Table 206 — WABEREQ field**

| Code | Description |
|---|---|
| 00b | Not specified. |
| 01b | The device server completes the read command specifying that LBA with GOOD status and any data transferred to the Data-In Buffer is indeterminate. |
| 10b | The device server terminates the read command specifying that LBA with CHECK CONDITION status with sense key set to MEDIUM ERROR and the additional sense code set to an appropriate value other than WRITE AFTER SANITIZE REQUIRED (e.g., ID CRC OR ECC ERROR). |
| 11b | The device server terminates the read command specifying that LBA with CHECK CONDITION status with sense key set to MEDIUM ERROR and the additional sense code set to WRITE AFTER SANITIZE REQUIRED. |

If a write medium operation has not been performed on a logical block referenced by a mapped LBA since a sanitize cryptographic erase operation was performed, and no other error occurs during the processing of a read command specifying that LBA, then the write after cryptographic erase required (WACEREQ) field indicates the device server behavior (see table 207).

**Table 207 — WACEREQ field**

| Code | Description |
|---|---|
| 00b | Not specified. |
| 01b | The device server completes the read command specifying that LBA with GOOD status and any data transferred to the Data-In Buffer is indeterminate. |
| 10b | The device server terminates the read command specifying that LBA with CHECK CONDITION status with sense key set to MEDIUM ERROR and the additional sense code set to an appropriate value other than WRITE AFTER SANITIZE REQUIRED (e.g., ID CRC OR ECC ERROR). |
| 11b | The device server terminates the read command specifying that LBA with CHECK CONDITION status with sense key set to MEDIUM ERROR and the additional sense code set to WRITE AFTER SANITIZE REQUIRED. |

The NOMINAL FORM FACTOR field indicates the nominal form factor of the device containing the logical unit and is defined in table 208.

**Table 208 — NOMINAL FORM FACTOR field**

| Code | Description |
|---|---|
| 0h | Nominal form factor is not reported |
| 1h | 5.25 inch |
| 2h | 3.5 inch |
| 3h | 2.5 inch |
| 4h | 1.8 inch |
| 5h | Less than 1.8 inch |
| All others | Reserved |

A VBULS (verify byte check unmapped LBA supported) bit set to one indicates that the device server supports unmapped LBAs while processing VERIFY commands (see 5.29, 5.30, 5.31, and 5.32) and WRITE AND VERIFY commands (see 5.37, 5.38, 5.39, and 5.40) with the BYTCHK field set to 01b. A VBULS bit set to zero indicates that the device server does not support unmapped LBAs while processing VERIFY commands and WRITE AND VERIFY commands with the BYTCHK field set to 01b. The device server should set the VBULS field to 01b.

### 6.6.3 Block Limits VPD page

The Block Limits VPD page (see table 209) provides the application client with the means to obtain certain operating parameters of the logical unit.

**Table 209 — Block Limits VPD page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PERIPHERAL QUALIFIER | | | PERIPHERAL DEVICE TYPE | | | | |
| 1 | PAGE CODE (B0h) | | | | | | | |
| 2 | (MSB) | | | PAGE LENGTH (003Ch) | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | Reserved | | | | | | | WSNZ |
| 5 | MAXIMUM COMPARE AND WRITE LENGTH | | | | | | | |
| 6 | (MSB) | | | OPTIMAL TRANSFER LENGTH GRANULARITY | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | (MSB) | | | MAXIMUM TRANSFER LENGTH | | | | |
| ••• | | | | | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | (MSB) | | | OPTIMAL TRANSFER LENGTH | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | (LSB) |
| 16 | (MSB) | | | MAXIMUM PREFETCH LENGTH | | | | |
| ••• | | | | | | | | |
| 19 | | | | | | | | (LSB) |
| 20 | (MSB) | | | MAXIMUM UNMAP LBA COUNT | | | | |
| ••• | | | | | | | | |
| 23 | | | | | | | | (LSB) |
| 24 | (MSB) | | | MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT | | | | |
| ••• | | | | | | | | |
| 27 | | | | | | | | (LSB) |
| 28 | (MSB) | | | OPTIMAL UNMAP GRANULARITY | | | | |
| ••• | | | | | | | | |
| 31 | | | | | | | | (LSB) |
| 32 | UGAVALID | (MSB) | | UNMAP GRANULARITY ALIGNMENT | | | | |
| ••• | | | | | | | | |
| 35 | | | | | | | | (LSB) |
| 36 | (MSB) | | | MAXIMUM WRITE SAME LENGTH | | | | |
| ••• | | | | | | | | |
| 43 | | | | | | | | (LSB) |
| 44 | | | | Reserved | | | | |
| ••• | | | | | | | | |
| 63 | | | | | | | | |

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in SPC-4.

The PAGE CODE field and PAGE LENGTH field are defined in SPC-4 and shall be set to the values shown in table 209.

A write same no zero (WSNZ) bit set to one indicates that the device server does not support a value of zero in the NUMBER OF LOGICAL BLOCKS field in the WRITE SAME command CDBs (see 5.43, 5.44, and 5.45). A WSNZ bit set to zero indicates that the device server may or may not support a value of zero in the NUMBER OF LOGICAL BLOCKS field of the WRITE SAME commands.

A MAXIMUM COMPARE AND WRITE LENGTH field set to a non-zero value indicates the maximum value that the device server accepts in the NUMBER OF LOGICAL BLOCKS field in the COMPARE AND WRITE command (see 5.2). A MAXIMUM COMPARE AND WRITE LENGTH field set to zero indicates that the device server does not support the COMPARE AND WRITE command. The device server shall set the MAXIMUM COMPARE AND WRITE LENGTH field to a value less than or equal to the value in the MAXIMUM TRANSFER LENGTH field.

An OPTIMAL TRANSFER LENGTH GRANULARITY field set to a non-zero value indicates the optimal transfer length granularity size in logical blocks for a single command shown in the command column of table 210. If a device server receives one of these commands with a transfer size that is not equal to a multiple of this value, then the device server may incur significant delays in processing the command. An OPTIMAL TRANSFER LENGTH GRANULARITY field set to 0000_0000h indicates that the device server does not report optimal transfer length granularity.

**Table 210 — Transfer limits for commands**

<table>
<tr><th>Command</th><th>Field that specifies the transfer size</th><th>Block Limits VPD page field(s) that indicate maximum limits</th><th>Additional sense code if the value in the specified field exceeds the maximum limit</th></tr>
<tr><td>COMPARE AND WRITE</td><td>CDB NUMBER OF LOGICAL BLOCKS field</td><td>MAXIMUM COMPARE AND WRITE LENGTH field</td><td></td></tr>
<tr><td>ORWRITE (16) / (32)</td><td>CDB TRANSFER LENGTH field</td><td>MAXIMUM TRANSFER LENGTH field</td><td rowspan="11">INVALID FIELD IN CDB</td></tr>
<tr><td>PRE-FETCH (10) / (16)</td><td>CDB PREFETCH LENGTH field</td><td>MAXIMUM PREFETCH LENGTH field</td></tr>
<tr><td>READ (10) / (12) / (16) / (32)</td><td>CDB TRANSFER LENGTH field</td><td rowspan="6">MAXIMUM TRANSFER LENGTH field</td></tr>
<tr><td>VERIFY (10) / (12) / (16) / (32)</td><td>CDB VERIFICATION LENGTH field</td></tr>
<tr><td>WRITE (10) / (12) / (16) / (32)</td><td rowspan="4">CDB TRANSFER LENGTH field</td></tr>
<tr><td>WRITE AND VERIFY (10) / (12) / (16) / (32)</td></tr>
<tr><td>XDWRITEREAD (10) / (32)</td></tr>
<tr><td>XPWRITE (10) / (32)</td></tr>
<tr><td>Any individual block device range descriptor in a POPULATE TOKEN command (see 5.7)</td><td rowspan="2">Block device range descriptor NUMBER OF LOGICAL BLOCKS field</td><td rowspan="2">MAXIMUM TRANSFER LENGTH field</td><td rowspan="2">INVALID FIELD IN PARAMETER LIST</td></tr>
<tr><td>Any individual block device range descriptor in a WRITE USING TOKEN command (see 5.46)</td></tr>
</table>

A MAXIMUM TRANSFER LENGTH field set to a non-zero value indicates the maximum transfer length in logical blocks that the device server accepts for a single command shown in table 210. If a device server receives one of these commands with a transfer size greater than this value, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to the value shown in table 210. A MAXIMUM TRANSFER LENGTH field set to 0000_0000h indicates that the device server does not report a limit on the transfer length.

An OPTIMAL TRANSFER LENGTH field set to a non-zero value indicates the optimal transfer size in logical blocks for a single command shown in table 210. If a device server receives one of these commands with a transfer size greater than this value, then the device server may incur significant delays in processing the command. An OPTIMAL TRANSFER LENGTH field set to 0000_0000h indicates that the device server does not report an optimal transfer size.

The MAXIMUM PREFETCH LENGTH field indicates the maximum prefetch length in logical blocks that the device server accepts for a single PRE-FETCH command.

The device server should set the MAXIMUM PREFETCH LENGTH field to less than or equal to the MAXIMUM TRANSFER LENGTH field.

A MAXIMUM UNMAP LBA COUNT field set to a non-zero value indicates the maximum number of LBAs that may be unmapped by an UNMAP command (see 5.28). If the number of LBAs that may be unmapped by an UNMAP command is constrained only by the amount of data that may be contained in the UNMAP parameter list (see 5.28.2), then the device server shall set the MAXIMUM UNMAP LBA COUNT field to FFFF_FFFFh. If the device server implements the UNMAP command, then the value in this field shall be greater than or equal to one. A MAXIMUM UNMAP LBA COUNT field set to 0000_0000h indicates that the device server does not implement the UNMAP command.

A MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT field set to a non-zero value indicates the maximum number of UNMAP block descriptors (see 5.28.2) that shall be contained in the parameter data transferred to the device server for an UNMAP command (see 5.28). If there is no limit on the number of UNMAP block descriptors contained in the parameter data, then the device server shall set the MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT field to FFFF_FFFFh. If the device server implements the UNMAP command, then the value in the MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT field shall be greater than or equal to one. A MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT field set to 0000_0000h indicates that the device server does not implement the UNMAP command.

An OPTIMAL UNMAP GRANULARITY field set to a non-zero value indicates the optimal granularity in logical blocks for unmap requests (e.g., an UNMAP command or a WRITE SAME (16) command with the UNMAP bit set to one). An unmap request with a number of logical blocks that is not a multiple of this value may result in unmap operations on fewer LBAs than requested. An OPTIMAL UNMAP GRANULARITY field set to 0000_0000h indicates that the device server does not report an optimal unmap granularity.

An unmap granularity alignment valid (UGAVALID) bit set to one indicates that the UNMAP GRANULARITY ALIGNMENT field is valid. A UGAVALID bit set to zero indicates that the UNMAP GRANULARITY ALIGNMENT field is not valid.

The UNMAP GRANULARITY ALIGNMENT field indicates the LBA of the first logical block to which the OPTIMAL UNMAP GRANULARITY field applies. The unmap granularity alignment is used to calculate an optimal unmap request starting LBA as follows:

optimal unmap request starting LBA = (n × optimal unmap granularity) + unmap granularity alignment

where:

| | |
|---|---|
| n | is zero or any positive integer value |
| optimal unmap granularity | is the value in the OPTIMAL UNMAP GRANULARITY field |
| unmap granularity alignment | is the value in the UNMAP GRANULARITY ALIGNMENT field |

An unmap request with a starting LBA that is not optimal may result in unmap operations on fewer LBAs than requested.

A MAXIMUM WRITE SAME LENGTH field set to a non-zero value indicates the maximum value that the device server accepts in the NUMBER OF LOGICAL BLOCKS field for a WRITE SAME command. A MAXIMUM WRITE SAME LENGTH field set to zero indicates that the device server does not report a limit on the number of logical blocks that may be requested for a single WRITE SAME command.

### 6.6.4 Logical Block Provisioning VPD page

The Logical Block Provisioning VPD page (see table 211) provides the application client with logical block provisioning related operating parameters of the logical unit.

**Table 211 — Logical Block Provisioning VPD page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PERIPHERAL QUALIFIER | | | PERIPHERAL DEVICE TYPE | | | | |
| 1 | PAGE CODE (B2h) | | | | | | | |
| 2 | (MSB) | | | PAGE LENGTH (n - 3) | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | THRESHOLD EXPONENT | | | | | | | |
| 5 | LBPU | LBPWS | LBPWS10 | Reserved | | LBPRZ | ANC_SUP | DP |
| 6 | Reserved | | | | | PROVISIONING TYPE | | |
| 7 | Reserved | | | | | | | |
| 8 | PROVISIONING GROUP DESCRIPTOR (if any) | | | | | | | |
| ••• | | | | | | | | |
| n | | | | | | | | |

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in SPC-4.

The PAGE CODE field is defined in SPC-4 and shall be set to the value shown in table 211.

The PAGE LENGTH field is defined in SPC-4. If the DP bit is set to zero, then the page length shall be set to 0004h. If the DP bit is set to one, then the page length shall be set to the value defined in table 211.

The THRESHOLD EXPONENT field indicates the threshold set size in LBAs as a power of 2 (i.e., the threshold set size is equal to $2^{(\text{threshold exponent})}$). A THRESHOLD EXPONENT field set to zero indicates that the logical unit does not support logical block provisioning thresholds.

If logical block provisioning thresholds are supported, then the threshold exponent shall be a non-zero value selected such that:

$$(\text{capacity} \div 2^{(\text{threshold exponent})}) < 2^{(32)}$$

where:

capacity — is 1 + the LBA of the last logical block as returned in the READ CAPACITY (16) parameter data (see 5.16.2) (i.e., the number of logical blocks on the direct access block device).

$2^{(32)}$ — is the constant value 1_0000_0000h (i.e., 4 294 967 296).

An LBPU bit set to one indicates that the device server supports the UNMAP command (see 5.28). An LBPU bit set to zero indicates that the device server does not support the UNMAP command.

An LBPWS bit set to one indicates that the device server supports the use of the WRITE SAME (16) command (see 5.44) to unmap LBAs. An LBPWS bit set to zero indicates that the device server does not support the use of the WRITE SAME (16) command to unmap LBAs.

An LBPWS10 bit set to one indicates that the device server supports the use of the WRITE SAME (10) command (see 5.43) to unmap LBAs. An LBPWS10 bit set to zero indicates that the device server does not support the use of the WRITE SAME (10) command to unmap LBAs.

The LBPRZ bit shall be set to the same value as the LBPRZ bit in the READ CAPACITY (16) parameter data (see 5.16.2).

An ANC_SUP bit set to one indicates that the device server supports anchored LBAs (see 4.7.1). An ANC_SUP bit set to zero indicates that the device server does not support anchored LBAs.

A descriptor present (DP) bit set to one indicates a PROVISIONING GROUP DESCRIPTOR field is present. A DP bit set to zero indicates a PROVISIONING GROUP DESCRIPTOR field is not present.

The PROVISIONING TYPE field is described in table 212.

**Table 212 — PROVISIONING TYPE field**

| Code | Description |
|---|---|
| 000b | The device server does not report a provisioning type. |
| 001b | The logical unit is resource provisioned (see 4.7.3.2). |
| 010b | The logical unit is thin provisioned (see 4.7.3.3). |
| All others | Reserved |

The PROVISIONING GROUP DESCRIPTOR field, if any, contains a designation descriptor (see SPC-4) for the LBA mapping resources used by this logical unit.

If a PROVISIONING GROUP DESCRIPTOR field is present:

a) the ASSOCIATION field shall be set to 00b (i.e. logical unit); and
b) the DESIGNATOR TYPE field shall be set to:
   A) 1h (i.e., T10 vendor ID based); or
   B) 3h (i.e., NAA).

### 6.6.5 Referrals VPD page

The Referrals VPD page (see table 213) contains parameters indicating characteristics of the user data segments contained within this logical unit.

**Table 213 — Referrals VPD page**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PERIPHERAL QUALIFIER | | | PERIPHERAL DEVICE TYPE | | | | |
| 1 | PAGE CODE (B3h) | | | | | | | |
| 2 | (MSB) | | | PAGE LENGTH (000Ch) | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | | | | Reserved | | | | |
| ••• | | | | | | | | |
| 7 | | | | | | | | |
| 8 | (MSB) | | | USER DATA SEGMENT SIZE | | | | |
| ••• | | | | | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | (MSB) | | | USER DATA SEGMENT MULTIPLIER | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | (LSB) |

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in SPC-4.

The PAGE CODE field and PAGE LENGTH field are defined in SPC-4 and shall be set to the values shown in table 213.

The PAGE LENGTH field is defined in SPC-4. The page length shall be set to the value shown in table 213.

A USER DATA SEGMENT SIZE field indicates the number of contiguous logical blocks in a user data segment (see 4.28.2). A USER DATA SEGMENT SIZE field set to zero indicates the user data segment size information (i.e., the first user data segment LBA to the last user data segment LBA) is as indicated in the user data segment referral descriptor (see table 14).

The USER DATA SEGMENT MULTIPLIER field is used by an application client to calculate the beginning LBA of each user data segment (see 4.28.2).

### 6.6.6 Third-Party Copy VPD page

#### 6.6.6.1 Third-Party Copy VPD page overview

The Third-Party Copy VPD page (see SPC-4) provides a means to retrieve third-party copy descriptors including a descriptor that describes operating parameters for the POPULATE TOKEN command (see 5.7) and the WRITE USING TOKEN command (see 5.46).

#### 6.6.6.2 Block device third-party copy descriptor type codes

Block device third-party copy descriptor type codes (see table 202) indicate which third-party copy descriptor is being returned.

**Table 214 — Block device third-party copy descriptor type codes**

| Third-party copy descriptor name | Page code | Reference | Support requirements |
|---|---|---|---|
| Block Device ROD Token Limits | 0000h | 6.6.6.3 | Optional [a] |
| See SPC-4 | All other codes | | |

[a] Mandatory if the POPULATE TOKEN command and the WRITE USING TOKEN command are supported.

### 6.6.6.3 Block Device ROD Token Limits descriptor

The Block Device ROD Token Limits descriptor (see table 215) is a third-party copy descriptor in the Third-Party Copy VPD page (see SPC-4) and provides the application client with a method to obtain certain operating parameters for block device ROD token operations (see 4.30).

**Table 215 — Block Device ROD Token Limits descriptor**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | THIRD-PARTY COPY DESCRIPTOR TYPE (0000h) | | | | |
| 1 | | | | | | | | (LSB) |
| 2 | (MSB) | | | THIRD-PARTY COPY DESCRIPTOR LENGTH (0020h) | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | | | | Vendor specific | | | | |
| ••• | | | | | | | | |
| 9 | | | | | | | | |
| 10 | (MSB) | | | MAXIMUM RANGE DESCRIPTORS | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | (MSB) | | | MAXIMUM INACTIVITY TIMEOUT | | | | |
| ••• | | | | | | | | |
| 15 | | | | | | | | (LSB) |
| 16 | (MSB) | | | DEFAULT INACTIVITY TIMEOUT | | | | |
| ••• | | | | | | | | |
| 19 | | | | | | | | (LSB) |
| 20 | (MSB) | | | MAXIMUM TOKEN TRANSFER SIZE | | | | |
| ••• | | | | | | | | |
| 27 | | | | | | | | (LSB) |
| 28 | (MSB) | | | OPTIMAL TRANSFER COUNT | | | | |
| ••• | | | | | | | | |
| 35 | | | | | | | | (LSB) |

The THIRD-PARTY COPY DESCRIPTOR TYPE field and the THIRD-PARTY COPY DESCRIPTOR LENGTH field are defined in SPC-4 and shall be set to the values show in table 215.

The MAXIMUM RANGE DESCRIPTORS field indicates the maximum number of block device range descriptors that may be specified in the parameter data of a POPULATE TOKEN command (see 5.7) and the parameter data of a WRITE USING TOKEN command (see 5.46). If the MAXIMUM RANGE DESCRIPTORS field is set to zero, then the copy manager does not report a maximum number of block device range descriptors.

The MAXIMUM INACTIVITY TIMEOUT field indicates the maximum value in the INACTIVITY TIMEOUT field of the parameter data of a POPULATE TOKEN command that is accepted by the copy manager. If the MAXIMUM INACTIVITY TIMEOUT field is set to zero, then the device server does not report a maximum inactivity timeout value. If the MAXIMUM INACTIVITY TIMEOUT field is set to FFFF_FFFFh then there is no maximum value that may be specified in the INACTIVITY TIMEOUT field in the parameter data of the POPULATE TOKEN command.

The DEFAULT INACTIVITY TIMEOUT field indicates the inactivity timeout value that is used if the INACTIVITY TIMEOUT field in the parameter data of a POPULATE TOKEN command is set to zero. If the DEFAULT INACTIVITY TIMEOUT field is set to zero, then the copy manager does not report a default inactivity timeout value.

The MAXIMUM TOKEN TRANSFER SIZE field indicates the maximum size in blocks that may be specified by:

a) the sum of the NUMBER OF LOGICAL BLOCKS fields in all block device range descriptors of the POPULATE TOKEN command; and
b) the sum of the NUMBER OF LOGICAL BLOCKS fields in all block device range descriptors of the WRITE USING TOKEN command.

If the MAXIMUM TOKEN TRANSFER SIZE field is set to zero, then the copy manager does not report a maximum token transfer size.

If the MAXIMUM BYTES IN BLOCK ROD field in the Block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page (see SPC-4) is reported, then the MAXIMUM TOKEN TRANSFER SIZE field shall be set to the number of logical blocks that represents the value in the MAXIMUM BYTES IN BLOCK ROD field in the Block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page.

The OPTIMAL TRANSFER COUNT field indicates the optimal number of blocks that the copy manager is able to transfer. If the sum of the NUMBER OF LOGICAL BLOCKS fields in all block device range descriptors in the parameter data of a POPULATE TOKEN command or the parameter data of a WRITE USING TOKEN command exceeds this value then, a significant delay in processing the request may be incurred. If the field is set to zero, then the copy manager does not report an optimal transfer count.

If the OPTIMAL BYTES IN BLOCK ROD TRANSFER field in the Block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page is reported, then the OPTIMAL TRANSFER COUNT field shall be set to the number of logical blocks that represents the value in the OPTIMAL BYTES IN BLOCK ROD TRANSFER field in the Block ROD device type specific features descriptor in the ROD token features third-party copy descriptor in the Third-party Copy VPD page.

### 6.7 Copy manager parameters

The copy manager parameters are device type specific data for ROD tokens (see SPC-4) created by a copy manager for a direct access block device (see 4.30) and are shown in table 216.

**Table 216 — ROD token device type specific data**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| ••• | LOGICAL BLOCK LENGTH IN BYTES | | | | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | Reserved | | | | P_TYPE | | | PROT_EN |
| 5 | P_I_EXPONENT | | | | LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT | | | |
| 6 | LBPME | LBPRZ | (MSB) | | | | | |
| 7 | LOWEST ALIGNED LOGICAL BLOCK ADDRESS | | | | | | | (LSB) |
| 8 | | | | | | | | |
| ••• | Reserved | | | | | | | |
| 31 | | | | | | | | |

The LOGICAL BLOCK LENGTH IN BYTES field, the P_TYPE field, the PROT_EN bit, the P_I_EXPONENT field, the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field, the LBPME bit, the LBPRZ bit, and the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field are defined in 5.16.2.

# Annex A
(informative)

# Numeric order codes

## A.1 Variable length CDBs

Some commands in table 31 (see 5.1) use the variable length command format defined in SPC-4. These commands use operation code 7Fh and are differentiated by service action codes as described in table A.1.

**Table A.1 — Variable length command service action code assignments**

| Operation code/service action code | Description |
|---|---|
| 7Fh/0000h | Reserved |
| 7Fh/0001h | Reserved |
| 7Fh/0002h | Reserved |
| 7Fh/0003h | Obsolete |
| 7Fh/0004h | Obsolete |
| 7Fh/0005h | Reserved |
| 7Fh/0006h | XPWRITE (32) |
| 7Fh/0007h | XDWRITEREAD (32) |
| 7Fh/0008h | Reserved |
| 7Fh/0009h | READ (32) |
| 7Fh/000Ah | VERIFY (32) |
| 7Fh/000Bh | WRITE (32) |
| 7Fh/000Ch | WRITE AND VERIFY (32) |
| 7Fh/000Dh | WRITE SAME (32) |
| 7Fh/000Eh | ORWRITE (32) |
| 7Fh/000Fh to 07FFh | Reserved |
| 7Fh/0800h to FFFFh | See SPC-4 |

## A.2 Service action CDBs

Some commands in table 31 (see 5.1) are implemented as device-type specific service actions for the SERVICE ACTION IN (16) operation code 9Eh (see SPC-4). These commands are differentiated by service action codes as described in table A.2.

**Table A.2 — SERVICE ACTION IN (16) service actions**

| Operation code/service action code | Description |
|---|---|
| 9Eh/00h to 0Fh | Reserved for commands applicable to all device types (see SPC-4) |
| 9Eh/10h | READ CAPACITY (16) |
| 9Eh/11h | READ LONG (16) |
| 9Eh/12h | GET LBA STATUS |
| 9Eh/13h | REPORT REFERRALS |
| 9Eh/14h to 1Fh | Reserved |

Some commands in table 31 (see 5.1) are implemented as device-type specific service actions for the SERVICE ACTION OUT (16) operation code 9Fh (see SPC-4). These commands are differentiated by service action codes as described in table A.3.

**Table A.3 — SERVICE ACTION OUT (16) service actions**

| Operation code/service action code | Description |
|---|---|
| 9Fh/00h to 0Fh | Reserved for commands applicable to all device types (see SPC-4) |
| 9Fh/10h | Reserved |
| 9Fh/11h | WRITE LONG (16) |
| 9Fh/12h to 1Fh | Reserved |

# Annex B
(informative)

# XOR command examples

## B.1 XOR command examples overview

This annex provides XOR command examples in storage array controller supervised configurations.

## B.2 Update write operation

Figure B.1 shows a read-modify-write operation supervised by a storage array controller. The example uses a supervising storage array controller, a direct-access block device holding XOR-protected data (i.e., the data disk), and a direct-access block device holding check data (i.e., the parity disk). Three SCSI commands are used: an XDWRITE command, an XDREAD command, and an XPWRITE command. An XDWRITEREAD command may be used in place of any sequence of an XDWRITE command followed by an XDREAD command.

The supervising storage array controller begins by sending XOR-protected data to the data disk using an XDWRITE command.

The data disk reads old XOR-protected data from its medium, performs an XOR operation using the old XOR-protected data and the XOR-protected data from the supervising storage array controller, stores the resulting intermediate XOR data in its buffer memory, and writes the XOR-protected data from the supervising storage array controller to its medium. The supervising storage array controller reads the resulting intermediate XOR data from the buffer memory by sending the data disk an XDREAD command.

The supervising storage array controller makes the resulting intermediate XOR data (i.e., data read with the XDREAD command) available to the parity disk by sending an XPWRITE command. The parity disk performs an XOR operation using the intermediate XOR data and the XOR data in its buffer memory. The resulting new XOR data is written to the medium.

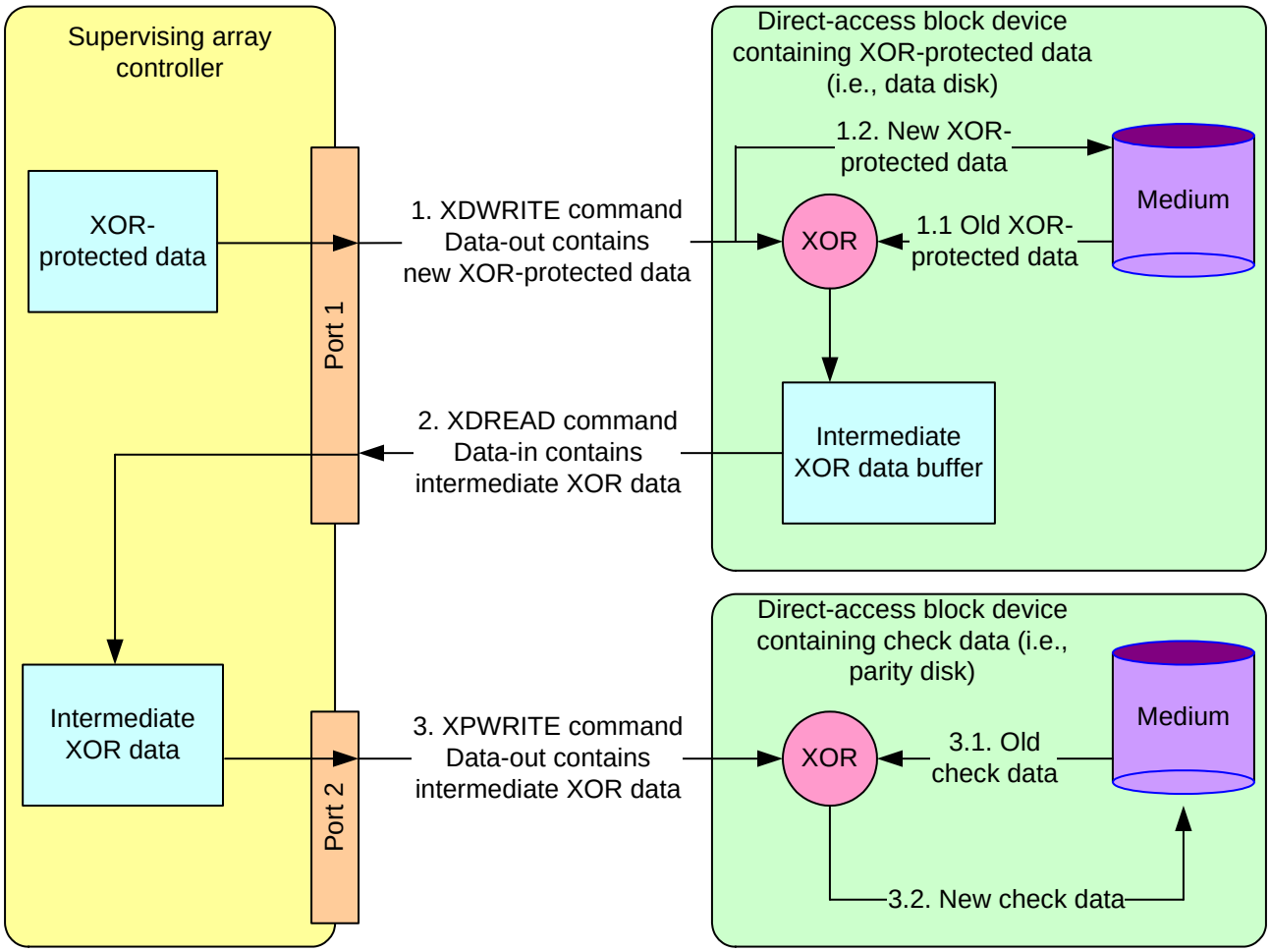



**Figure B.1 — Update write operation (storage array controller supervised)**

## B.3 Regenerate operation

Figure B.2 shows a regenerate operation supervised by a storage array controller. The example uses a supervising storage array controller and three direct-access block devices (i.e., disk 1, disk 2, and disk 3). Three SCSI commands are used: a READ command, an XDWRITE command, and an XDREAD command. An XDWRITEREAD command may be used in place of any sequence of an XDWRITE command followed by an XDREAD command.

The supervising storage array controller begins by issuing a READ command to disk 1. The data received from this command is sent by the supervising storage array controller to disk 2 using an XDWRITE command with the DISABLE WRITE bit set to one. Disk 2 reads data from its medium, performs an XOR operation using that data and the data received from the supervising storage array controller, and stores the resulting intermediate XOR data in its buffer memory. The supervising storage array controller retrieves the intermediate XOR data from the buffer memory by issuing an XDREAD command to disk 2. The supervising storage array controller issues XDWRITE commands and XDREAD commands in the same manner to disk 3.

The resulting data from disk 3 is the regenerated data.

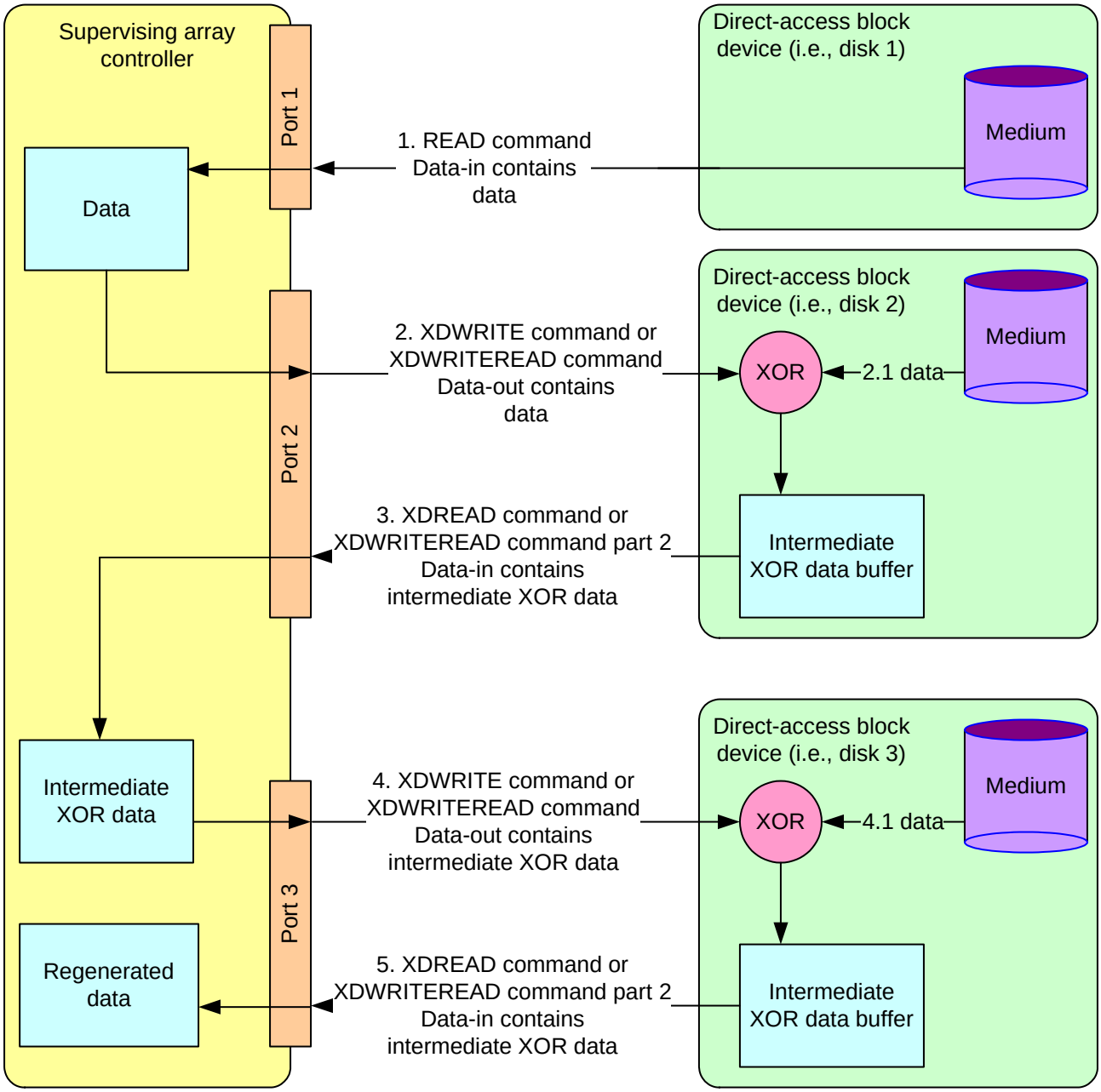



**Figure B.2 — Regenerate operation (storage array controller supervised)**

## B.4 Rebuild operation

Figure B.3 shows a rebuild operation supervised by a storage array controller. The example uses a supervising storage array controller, two direct-access block devices (i.e., disk 1 and disk 2) as the source devices, and one direct-access block device (i.e., disk 3) as the rebuild device. Four SCSI commands are used: a READ command, an XDWRITE command, an XDREAD command, and a WRITE command. An XDWRITEREAD command may be used in place of any sequence of an XDWRITE command followed by an XDREAD command.

The supervising storage array controller begins by issuing a READ command to disk 1. The data received from the READ command is sent by the supervising storage array controller to disk 2 using an XDWRITE command with a DISABLE WRITE bit set to one. Disk 2 reads data from its medium, performs an XOR operation using that data and the data received from the supervising storage array controller, and stores the resulting

intermediate XOR data in its buffer memory. The supervising storage array controller retrieves the intermediate XOR data by sending an XDREAD command to disk 2.

The resulting data from disk 2 is the rebuilt data and is sent to the direct-access block device being rebuilt (i.e., disk 3) using a WRITE command.

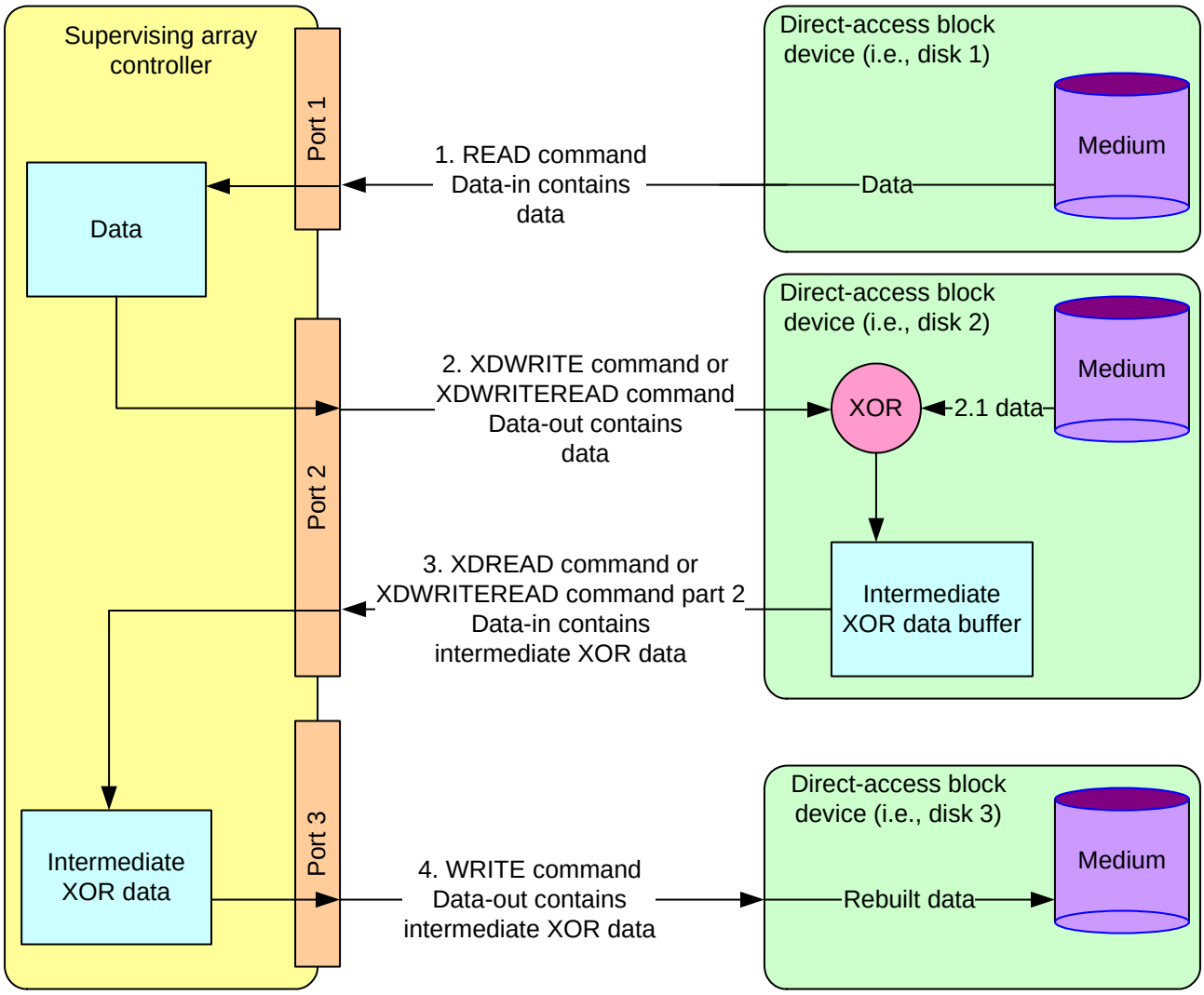



**Figure B.3 — Rebuild operation (storage array controller supervised)**

# Annex C
(informative)

## CRC example in C

The following is an example C program that generates the value for the LOGICAL BLOCK GUARD field in protection information (see 4.22).

```
// picrc.cpp : SCSI SBC-2 Protection Information CRC generator
#include “stdafx.h”
#include <stdio.h>
#include <malloc.h>

/* return crc value */
unsigned short calculate_crc(unsigned char *frame, unsigned long length) {
unsigned short const poly = 0x8BB7L;  /* Polynomial */
unsigned const int poly_length = 16;
unsigned short crc_gen;
unsigned short x;
unsigned int i, j, fb;
unsigned const int invert = 0;/* 1=seed with 1s and invert the CRC */

crc_gen = 0x0000;
crc_gen ^= invert? 0xFFFF: 0x0000;   /* seed generator */

for (i = 0; i < length; i += 2) {
      /* assume little endian */
      x = (frame[i] << 8) | frame[i+1];

      /* serial shift register implementation */
        for (j = 0; j < poly_length; j++) {
            fb = ((x & 0x8000L) == 0x8000L) ^ ((crc_gen & 0x8000L) ==
0x8000L);
            x <<= 1;
            crc_gen <<= 1;
            if (fb)
                  crc_gen ^= poly;
      }
}

return crc_gen ^ (invert? 0xFFFF: 0x0000); /* invert output */
} /* calculate_crc */

/* function prototype */
unsigned short calculate_crc(unsigned char *, unsigned long);

void main (void) {
unsigned char *buffer;
unsigned long buffer_size = 32;
unsigned short crc;
unsigned int i;

/* 32 0x00 */
buffer = (unsigned char *) malloc (buffer_size);
for (i = 0; i < buffer_size; i++) {
      buffer[i] = 0x00;
```

```
}
crc = calculate_crc(buffer, buffer_size);
printf (“Example CRC all-zeros is %04x\n”, crc);
free (buffer);

/* 32 0xFF */
buffer = (unsigned char *) malloc (buffer_size);
for (i = 0; i < buffer_size; i++) {
      buffer[i] = 0xFF;
}
crc = calculate_crc(buffer, buffer_size);
printf (“Example CRC all-ones is %04x\n”, crc);
free (buffer);

/* 0x00 incrementing to 0x1F */
buffer = (unsigned char *) malloc (buffer_size);
for (i = 0; i < buffer_size; i++) {
      buffer[i] = i;
}
crc = calculate_crc(buffer, buffer_size);
printf (“Example CRC incrementing is %04x\n”, crc);
free (buffer);

/* 0xFF 0xFF then 30 zeros */
buffer = (unsigned char *) malloc (buffer_size);
buffer[0] = 0xff;
buffer[1] = 0xff;
for (i = 2; i < buffer_size; i++) {
      buffer[i] = 0x00;
}
crc = calculate_crc(buffer, buffer_size);
printf (“Example CRC FF FF then 30 zeros is %04x\n”, crc);
free (buffer);

/* 0xFF decrementing to 0xE0 */
buffer = (unsigned char *) malloc (buffer_size);
for (i = 0; i < buffer_size; i++) {
      buffer[i] = 0xff - i;
}
crc = calculate_crc(buffer, buffer_size);
printf (“Example CRC FF decrementing to E0 is %04x\n”, crc);
free (buffer);

} /* main */
```

# Annex D
(informative)

## Sense information for locked or encrypted SCSI target devices

A device sever may complete some commands with CHECK CONDITION status under certain conditions when the SCSI target device is locked or encrypted. Table D.1 describes the conditions relative to the sense key and the additional sense code returned by the device server with the CHECK CONDITION status.

**Table D.1 — Sense information for locked or encrypted SCSI target devices**

| Sense key | Additional sense code | Description |
|---|---|---|
| DATA PROTECT | ACCESS DENIED – NO ACCESS RIGHTS | The SCSI target device is locked. This condition may occur for read medium operations, write medium operations, or both. This condition may occur for the entire SCSI target device or for a range of LBAs contained in the SCSI target device. To clear this condition, an application client performs a security protocol specific procedure to unlock access to the SCSI target device. |
| ABORTED COMMAND | LOGICAL BLOCK REFERENCE TAG CHECK FAILED | These conditions may occur for a read medium operation. The additional sense codes may indicate that an encrypting SCSI target device has changed the encryption/decryption key, and the LBAs requested by the command have not yet been rewritten. Disabling protection information checking in a CDB may allow the command to complete successfully, but the data returned for the command may be invalid (i.e., not decrypted). To clear this condition, an application client writes the LBAs for which the condition occurred with new data. |
| ABORTED COMMAND | LOGICAL BLOCK APPLICATION TAG CHECK FAILED | |
| ABORTED COMMAND | LOGICAL BLOCK GUARD CHECK FAILED | |

# Annex E
(informative)

# Optimizing block access characteristics

## E.1 Optimizing block access overview

This annex describes an example method that application clients may use to achieve optimal performance for logical block access. This example uses the following information:

a) the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field in the READ CAPACITY (16) parameter data (see 5.16.2);
b) the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field in the READ CAPACITY (16) parameter data;
c) the OPTIMAL TRANSFER LENGTH GRANULARITY field in the Block Limits VPD page (see 6.6.3);
d) the OPTIMAL TRANSFER LENGTH field in the Block Limits VPD page; and
e) the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page.

## E.2 Starting logical block offset

The READ CAPACITY (16) command transfers parameter data which includes a value in the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field. As shown in figure 4, the value in this field indicates the starting alignment of logical block addresses where optimal performance for logical block access begins.

## E.3 Optimal granularity sizes

The READ CAPACITY (16) command transfers parameter data that includes a value in the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field. As shown in figure 2 and in figure 4, the value in this field enables the application client to determine the number of logical blocks per physical block.

The Block Limits VPD page may include values in the OPTIMAL TRANSFER LENGTH GRANULARITY field, the OPTIMAL TRANSFER LENGTH field, and the MAXIMUM TRANSFER LENGTH field. These values may be used to determine optimum transfer sizes.

If the OPTIMAL TRANSFER LENGTH GRANULARITY field is valid (i.e., contains a value greater than zero), then the value in the OPTIMAL TRANSFER LENGTH GRANULARITY field is the optimal granularity size. If:

a) the Block Limits VPD page is not supported; or
b) the Block Limits VPD page is supported and the OPTIMAL TRANSFER LENGTH GRANULARITY field is set to zero,

then the value $2^{(\text{LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT})}$ is the optimal granularity size.

## E.4 Optimizing transfers

To obtain optimal performance, the application client requests transfers with a starting LBA of the form calculated by the following formula:

starting LBA = lowest aligned LBA + (optimal transfer length granularity × n)

where:

| | |
|---|---|
| starting LBA | is the LBA of the first logical block accessed |
| lowest aligned LBA | is the value in the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field |
| n | is zero or a positive integer |

and using transfer lengths of the form:

transfer length = (optimal granularity size × k)

where:

| | |
|---|---|
| transfer length | is the number of contiguous logical blocks of data being accessed |
| optimal granularity size | is the value described in E.3 |
| k | is a positive integer |

To obtain optimal performance, the application client requests a transfer length, in logical blocks, that is no larger than the value in the MAXIMUM TRANSFER LENGTH field, and is:

a) no larger than the optimal transfer length for devices where the delay in processing transfers larger than the optimal transfer length is large; or
b) not limited by the value in the OPTIMAL TRANSFER LENGTH field for devices where the delay in processing transfers larger than the optimal transfer length is small (i.e., most direct access block devices exhibit this type of operation).

NOTE 1 - There is no method available to determine if the significant delay in processing for various transfer lengths is large or small.

NOTE 2 - To achieve optimum performance, it is more important that the application client meet the device's starting and ending alignment boundary conditions than the maximum transfer length conditions. These considerations have larger impacts on write performance than read performance.

## E.5 Examples

In this example, a logical unit reports the following information:

a) the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 0003h in the READ CAPACITY (16) parameter data (see 5.16.2);
b) the OPTIMAL TRANSFER LENGTH GRANULARITY field set to 0008h in the Block Limits VPD page (see 6.6.3);
c) the MAXIMUM TRANSFER LENGTH field set to 0000_0000h in the Block Limits VPD page; and
d) the OPTIMAL TRANSFER LENGTH field set to 0000_0080h (i.e., 128) in the Block Limits VPD page.

The starting LBA for optimal transfers on this logical unit should be of the form ((8 × n) + 3) (e.g., starting LBAs of 3, 11, 19, 27, and 35). The transfer length for optimal transfers should be a multiple of 8 logical blocks (e.g., transfer lengths of 8 blocks, 32 blocks, or 128 blocks).

A write command with the LBA field set to 19 and the transfer length field set to 32 should exhibit improved performance over a write command with the LBA field set to 18 and the transfer length field set to 32.

If the device has a significant delay in processing transfers larger than the optimal transfer length, some operations may exhibit improved performance if a single large request is broken info multiple smaller requests (e.g., rather than performing a single read of 248 logical blocks, the transfer may be optimized by setting the transfer length of one read command to 128 logical blocks and setting the transfer length of a second read command to 120 logical blocks).

# Annex F
(informative)

# Logical block provisioning reporting examples

## F.1 Logical block provisioning reporting examples overview

Logical block provisioning reporting may be implemented using different methods. Implementations may include one or more of the following:

a) use of dedicated LBA mapping resources (e.g., resources are associated with a specific logical unit);
a) use of shared LBA mapping resources (e.g., resources are shared by multiple logical units);
b) reporting based on dedicated LBA mapping resources (e.g., resources are reported specific to the logical unit);
c) reporting based on shared LBA mapping resources (e.g., resources are reported for the resource pool as a whole);
d) LBA mapping resource tracking based on logical blocks; and
e) LBA mapping resource tracking based on threshold sets.

This annex describes examples of logical block provisioning reporting. Each example follows logical block provisioning resource usage and reporting over time as a specified set of operations occur.

## F.2 Interpreting log parameter counts

Due to the variation of the threshold set size implementations, logical block usage and resource reporting may not have a direct relationship. The second example (see F.4) demonstrates an implementation where logical blocks are allocated on an individual basis, and reported using a larger threshold set basis. The reporting is a direct calculation from a logical block based count to a threshold set based count.

In implementations where a threshold set contains a set of contiguous logical blocks, the reporting may be substantially different. LUN 1 in the first example (see F.3) demonstrates such an implementation. At the initial conditions, two threshold sets are reported as being used. With a threshold set size of 1 024 blocks, these two threshold sets may contain as little as one logical block of application client data in each threshold set, or as many as 1 024 contiguous logical blocks in each threshold set. Which LBAs have been written by the application client will have a substantial impact on how the usage of those resources is reported.

The relationship of the physical blocks to the logical blocks (see figure 4 and figure 5) may have an impact on the logical block provisioning log parameters. Which LBAs are written by the application may impact the number of physical blocks required to be allocated and therefore impact the reporting of the LBA mapping resource parameters.

The device server may not prioritize the maintenance of the values in this log page above the completion of other operations (e.g., read medium operations or write medium operations). This may result in delays in updates to these values (e.g., after a request to unmap a large number of logical blocks). The logical block provisioning log parameters may also appear inaccurate for devices that perform unmap operations using a periodic background function.

As a result, application clients using logical block provisioning thresholds and examining logical block provisioning log parameters should not expect application client determined usage values or application client determined available space values to match log parameters or threshold events as reported by the logical unit.

## F.3 Dedicated resource, threshold set tracked example

### F.3.1 Dedicated resource, threshold set tracked example overview

This example describes a method that reports dedicated logical block provisioning resources based on threshold sets. In this example, the values reported by the logical unit in the Logical Block Provisioning log page (see 6.4.4) reflect the usage for each logical unit, and the available resources dedicated to each logical unit. Each threshold set is allocated to contain a set of contiguous logical blocks (e.g., LBAs 1024 to 2047 are contained in the same threshold set).

### F.3.2 Dedicated resource, threshold set tracked example configuration

The configuration used for this example consists of two logical units, each with dedicated logical block provisioning resources. Table F.1 shows logical block provisioning related capacity values used in this example.

**Table F.1 — Dedicated resource, threshold set tracked example capacity information**

| LUN | Capacity [a] | THRESHOLD EXPONENT field [b] | Number of threshold sets [c] |
|---|---|---|---|
| 1 | 4000_0000h (i.e., 1 GiB) | 0Ah (i.e., 512 KiB, 1 024 logical blocks) | 0010_0000h (i.e., 2 Mi) |
| 2 | C000_0000h (i.e., 3 GiB) | 0Ch (i.e., 2 MiB, 4 096 logical blocks) | 000C_0000h (i.e., 768 Ki) |

[a] RETURNED LOGICAL BLOCK ADDRESS field in READ CAPACITY parameter data (see 5.15.2 and 5.16.2).
[b] In the Logical Block Provisioning VPD page (see 6.6.4).
[c] Number of threshold sets = capacity ÷ $2^{(\text{threshold exponent})}$.

Table F.2 shows LUN 1 with four enabled threshold descriptors and LUN 2 with two enabled threshold descriptors. The threshold descriptors in the Logical Block Provisioning mode page (see 6.5.7) for LUN 1 are configured to report a logical block provisioning threshold crossing (see 4.7.3.8) when:

a) available LBA mapping resources reaches 30% of reported capacity;
b) available LBA mapping resources reaches 20% of reported capacity;
c) available LBA mapping resources reaches 10% of reported capacity; or
d) used LBA mapping resources reaches 75% of reported capacity.

The threshold descriptors in the Logical Block Provisioning mode page for LUN 2 are configured to report a logical block provisioning threshold crossing when:

a) available LBA mapping resources reaches 50% of reported capacity; or
b) available LBA mapping resources reaches 10% of reported capacity.

**Table F.2 — Dedicated resource, threshold set tracked example capacity information**

| LUN | Threshold resource [a] | Threshold count [b] | Description |
|---|---|---|---|
| 1 | 0001h | 0004_CCCCh | An available LBA mapping resource threshold set to 30% of the reported capacity (i.e., number of threshold sets from table F.1 × 0.30 = 0004_CCCCh threshold sets). |
| | 0001h | 0003_3333h | An available LBA mapping resource threshold set to 20% of the reported capacity. |
| | 0001h | 0001_9999h | An available LBA mapping resource threshold set to 10% of the reported capacity. |
| | 0002h | 000C_0000h | A used LBA mapping resource threshold set to 75% of the reported capacity (i.e., number of threshold sets from table F.1 × 0.75 = 0007_5000h threshold sets). |
| 2 | 0001h | 0006_0000h | An available LBA mapping resource threshold set to 50% of the reported capacity (i.e., number of threshold sets from table F.1 × 0.50 = 0006_0000h threshold sets). |
| | 0001h | 0001_3333h | An available LBA mapping resource threshold set to 10% of the reported capacity. |

[a] THRESHOLD RESOURCE field in the Logical Block Provisioning mode page (see 6.5.7.2) and the PARAMETER CODE field in Logical Block Provisioning log page (see 6.4.4.2).
[b] THRESHOLD COUNT field in the Logical Block Provisioning mode page.

## F.3.3 Dedicated resource, threshold set tracked example sequence

The sequence of events for this example are:

1) Initial conditions (see F.3.4);
2) Operations that occur (see F.3.5); and
3) Final values in the logical block provisioning log page (see F.3.6).

## F.3.4 Dedicated resource, threshold set tracked example initial conditions

Initially, LUN 1 has 2 threshold sets used, and has 69 108 736 logical blocks available (i.e. 0001_07A1h threshold sets). The application client has written at least one logical block into each logical block range that corresponds to a threshold set, therefore the application client may have written from 2 logical blocks to 2048 logical blocks. LUN 2 has 1 073 741 824 logical blocks available (i.e., 0004_0000h threshold sets). LUN 2 does not report a used LBA mapping resource parameter. Table F.3 shows the values in the Logical Block Provisioning log page for the initial conditions in this example.

**Table F.3 — Dedicated resource, threshold set tracked example initial conditions**

| LUN | Log page parameter [a] | Resource count [b] | Scope [c] | Description [d] |
|---|---|---|---|---|
| 1 | 0001h | 0001_07A1h | 01b | The available LBA mapping resource parameter indicates that 69 108 736 logical blocks (i.e. 1_07A1h threshold sets × 1 024 logical blocks per threshold set) are available for LUN 1. |
| | 0002h | 0000_0002h | 01b | The used LBA mapping resource parameter indicates that 2 048 logical blocks (i.e., 2h threshold sets × 1 024 logical blocks per threshold set) have been used (i.e., allocated) by LUN 1. |
| 2 | 0001h | 0004_0000h | 01b | The available LBA mapping resource parameter indicates that 1 073 741 824 logical blocks (i.e., 4_0000h threshold sets × 4 096 logical blocks per threshold set) are available for LUN 2. |

[a] THRESHOLD RESOURCE field in the Logical Block Provisioning mode page (see 6.5.7.2) and the PARAMETER CODE field in Logical Block Provisioning log page (see 6.4.4.2).
[b] RESOURCE COUNT field in the Logical Block Provisioning mode page.
[c] SCOPE field in the Logical Block Provisioning log page.
[d] LBA count = capacity × $2^{(\text{threshold exponent})}$.

## F.3.5 Operations that occur

Write medium operations occur to LUN 1 that require one additional threshold set to be allocated when the application client writes 50 additional contiguous logical blocks. Used LBA mapping resources on LUN 1 are now 3 072 logical blocks (i.e., three threshold sets), and available LBA mapping resources are 69 107 712 logical blocks. Write medium operations also occur to LUN 2 that require no additional threshold sets when the application client writes an additional 100 logical blocks into a threshold set that was already allocated.

### F.3.6 Dedicated resource, threshold set tracked example final log page values

Table F.4 shows the values in the Logical Block Provisioning log page after the operations described in F.3.5 have occurred.

**Table F.4 — Dedicated resource, threshold set tracked example final log page values**

| LUN | Log page parameter [a] | Resource count [b] | Scope [c] | Description [d] |
|---|---|---|---|---|
| 1 | 0001h | 0001_07A1h | 01b | The available LBA mapping resource parameter indicates that 69 107 712 logical blocks (i.e. 1_07A0h threshold sets × 1 024 logical blocks per threshold set) are available for LUN 1. |
| | 0002h | 0000_0002h | 01b | The used LBA mapping resource parameter indicates that 3 072 logical blocks (i.e., 3h threshold sets × 1 024 logical blocks per threshold set) have been used (i.e., allocated) by LUN 1. |
| 2 | 0001h | 0004_0000h | 01b | The available LBA mapping resource parameter indicates that 1 073 741 824 (i.e., 4_0000h threshold sets × 4 096 logical blocks per threshold set) are available for LUN 2. |

[a] THRESHOLD RESOURCE field in the Logical Block Provisioning mode page (see 6.5.7.2) and the PARAMETER CODE field in Logical Block Provisioning log page (see 6.4.4.2).
[b] RESOURCE COUNT field in the Logical Block Provisioning mode page.
[c] SCOPE field in the Logical Block Provisioning log page.
[d] LBA count = capacity × $2^{(\text{threshold exponent})}$.

## F.4 Shared resource, logical block tracked example

### F.4.1 Shared resource, logical block tracked example overview

This example describes a method that tracks shared logical block provisioning resources based on logical blocks. The logical block provisioning resources are shared by multiple logical units. In this example, the values reported by the logical unit in the Logical Block Provisioning log page (see 6.4.4) reflect the combined usage of the logical units that share the logical block provisioning resources, and the resources available for use by any of the logical units that share the logical block provisioning resources. Resources are allocated one logical block at a time but reported with a larger threshold set size.

### F.4.2 Shared resource, logical block tracked example configuration

The configuration used for this example consists of two logical units, where the logical block provisioning resources are shared between both logical units. Table F.5 shows logical block provisioning related capacity values used in this example.

**Table F.5 — Shared resource, logical block tracked example capacity information**

| LUN | Capacity [a] | THRESHOLD EXPONENT field [b] | Number of threshold sets [c] |
|---|---|---|---|
| 1 | 4000_0000h (i.e., 1 GiB) | 0Bh (i.e., 1 MiB, 2 048 logical blocks) | 0008_0000h (i.e., 512 Ki) |
| 2 | C000_0000h (i.e., 3 GiB) | 0Bh (i.e., 1 MiB, 2 048 logical blocks) | 0018_0000h (i.e., 1 536 Ki) |

[a] RETURNED LOGICAL BLOCK ADDRESS field in READ CAPACITY parameter data (see 5.15.2 and 5.16.2).
[b] In the Logical Block Provisioning VPD page (see 6.6.4).
[c] Number of threshold sets = capacity ÷ $2^{(\text{threshold exponent})}$.

## F.4.3 Shared resource, logical block tracked example time line

The sequence of events for this example are:

1) Initial conditions (see F.4.4);
2) Operations that occur (see F.4.5); and
3) Final values in the logical block provisioning log page (see F.4.6).

## F.4.4 Shared resource, logical block tracked example initial conditions

Initially, LUN 1 and LUN 2 have used a combined total of 57 000 logical blocks. LUN1 and LUN 2 have 1 073 741 900 logical blocks available for use by either LUN 1 or LUN 2. Table F.6 shows the values in the Logical Block Provisioning log page for the initial conditions in this example.

**Table F.6 — Shared resource, logical block tracked example initial conditions**

| LUN | Log page parameter [a] | Resource count [b] | Scope [c] | Description |
|---|---|---|---|---|
| 1 | 0001h | 0008_0000h | 10b | The available LBA mapping resource parameter indicates that from 1 073 741 824 logical blocks (i.e. 8_0000h threshold sets × 2 048 logical block per threshold set) to 1 073 743 871 logical blocks are available for LUN 1 or LUN 2. [d] |
| | 0002h | 0000_001Ch | 10b | The used LBA mapping resource parameter indicates that from 55 297 logical blocks to 57 344 logical blocks (i.e., 1Ch threshold sets × 2 048 logical blocks per threshold set) have been used (i.e., allocated) by LUN 1 and LUN 2. [e] |
| 2 | 0001h | 0008_0000h | 10b | The available LBA mapping resource parameter indicates that 1 073 741 824 logical blocks (i.e., 4_0000h threshold sets × 4 096 logical blocks per threshold set) are available for LUN 2. [d] |

[a] THRESHOLD RESOURCE field in the Logical Block Provisioning mode page (see 6.5.7.2) and the PARAMETER CODE field in Logical Block Provisioning log page (see 6.4.4.2).
[b] RESOURCE COUNT field in the Logical Block Provisioning mode page.
[c] SCOPE field in the Logical Block Provisioning log page.
[d] Minimum available LBA count = resource count × $2^{(\text{threshold exponent})}$.
[e] Maximum used LBA count = resource count × $2^{(\text{threshold exponent})}$.

## F.4.5 Operations that occur

Write medium operations occur to LUN 1 that require 2 000 additional logical blocks to be used and write medium operations occur to LUN 2 that require 3 000 additional logical blocks to be used. Used LBA mapping resources on LUN 1 and LUN 2 are now 62 000 logical blocks, and the combined LBA mapping resources available to both LUN 1 and LUN 2 are 1 073 736 900 logical blocks (i.e., 1 073 741 900 minus 5 000).

## F.4.6 Shared resource, logical block tracked example final log page values

Table F.7 shows the values in the Logical Block Provisioning log page after the operations described in F.4.5 have occurred.

**Table F.7 — Shared resource, logical block tracked example final log page values**

| LUN | Log page parameter [a] | Resource count [b] | Scope [c] | Description |
|---|---|---|---|---|
| 1 | 0001h | 0007_FFFDh | 10b | The available LBA mapping resource parameter indicates that from 1 073 735 680 logical blocks (i.e. 7_FFFDh threshold sets × 2 048 logical blocks per threshold set) to 1 073 737 727 logical blocks are available for LUN 1 or LUN 2. [d] |
| | 0002h | 0000_001Fh | 10b | The used LBA mapping resource parameter indicates that from 61 441 logical blocks to 63 488 logical blocks (i.e., 1Fh threshold sets × 2 048 logical blocks per threshold set) have been used (i.e., allocated) by LUN 1 and LUN 2. [e] |
| 2 | 0001h | 0007_FFFDh | 10b | The available LBA mapping resource parameter indicates that from 1 073 735 680 logical blocks (i.e. 7_FFFDh threshold sets × 2 048 logical blocks per threshold set) to 1 073 737 727 logical blocks are available for LUN 1 or LUN 2. [d] |

[a] THRESHOLD RESOURCE field in the Logical Block Provisioning mode page (see 6.5.7.2) and the PARAMETER CODE field in Logical Block Provisioning log page (see 6.4.4.2).
[b] RESOURCE COUNT field in the Logical Block Provisioning mode page.
[c] SCOPE field in the Logical Block Provisioning log page.
[d] Minimum available LBA count = resource count × $2^{(\text{threshold exponent})}$.
[e] Maximum used LBA count = resource count × $2^{(\text{threshold exponent})}$.

## F.5 Shared available, dedicated used, logical block tracked example

### F.5.1 Shared available, dedicated used, logical block tracked example overview

This example describes a method that tracks available shared logical block provisioning resources based on logical blocks and dedicated used logical block provisioning resources based on logical blocks. The available logical block provisioning resources are shared by multiple logical units. In this example, the values reported by the logical unit in the available LBA mapping resource parameter of the Logical Block Provisioning log page (see 6.4.4) reflect the resources available for use by any of the logical units that share the logical block provisioning resources. The values reported by the logical unit in the used LBA mapping resource parameter of the Logical Block Provisioning log page reflect the usage for the individual logical unit.

### F.5.2 Shared available, dedicated used, logical block tracked example configuration

The configuration used for this example consists of two logical units, where the available logical block provisioning resources are shared between both logical units and used logical block provisioning resources are reported independently for each logical unit. Table F.5 shows logical block provisioning related capacity values used in this example.

**Table F.8 — Shared available, dedicated used example capacity information**

| LUN | Capacity [a] | THRESHOLD EXPONENT field [b] | Number of threshold sets [c] |
|---|---|---|---|
| 1 | 4000_0000h (i.e., 1 GiB) | 0Bh (i.e., 1 MiB, 2 048 logical blocks) | 0008_0000h (i.e., 512 Ki) |
| 2 | C000_0000h (i.e., 3 GiB) | 0Bh (i.e., 1 MiB, 2 048 logical blocks) | 0018_0000h (i.e., 1 536 Ki) |

[a] RETURNED LOGICAL BLOCK ADDRESS field in READ CAPACITY parameter data (see 5.15.2 and 5.16.2).
[b] In the Logical Block Provisioning VPD page (see 6.6.4).
[c] Number of threshold sets = capacity ÷ $2^{(\text{threshold exponent})}$.

### F.5.3 Shared available, dedicated used, logical block tracked example time line

The sequence of events for this example are:

1) Initial conditions (see F.5.4);
2) Operations that occur (see F.5.5); and
3) Final values in the logical block provisioning log page (see F.5.6).

## F.5.4 Shared available, dedicated used, logical block tracked example initial conditions

Initially, LUN 1 has used 57 000 logical blocks and, LUN 2 has used 103 000 logical blocks. LUN 1 and LUN 2 have 1 073 741 900 logical blocks available for use by either LUN 1 or LUN 2. Table F.9 shows the values in the Logical Block Provisioning log page for the initial conditions in this example.

**Table F.9 — Shared resource, logical block tracked example initial conditions**

| LUN | Log page parameter [a] | Resource count [b] | Scope [c] | Description |
|---|---|---|---|---|
| 1 | 0001h | 0008_0000h | 10b | The available LBA mapping resource parameter indicates that from 1 073 741 824 logical blocks (i.e. 8_0000h threshold sets × 2 048 logical blocks per threshold set) to 1 073 743 871 logical blocks are available for LUN 1 or LUN 2. [d] |
| | 0002h | 0000_001Ch | 01b | The used LBA mapping resource parameter indicates that from 55 297 logical blocks to 57 344 logical blocks (i.e., 1Ch threshold sets × 2 048 logical blocks per threshold set) have been used (i.e., allocated) by LUN 1. [e] |
| 2 | 0001h | 0008_0000h | 10b | The available LBA mapping resource parameter indicates that from 1 073 741 824 logical blocks (i.e., 8_0000h threshold sets × 2 048 logical blocks per threshold set) to 1 073 743 871 logical blocks are available for LUN 1 or LUN 2. [d] |
| | 0002h | 0000_0033h | 01b | The used LBA mapping resource parameter indicates that from 102 401 logical blocks to 104 448 (i.e., 33h threshold sets × 2 048 logical blocks per threshold set) have been used (i.e., allocated) by LUN 2. [e] |

[a] THRESHOLD RESOURCE field in the Logical Block Provisioning mode page (see 6.5.7.2) and the PARAMETER CODE field in Logical Block Provisioning log page (see 6.4.4.2).
[b] RESOURCE COUNT field in the Logical Block Provisioning mode page.
[c] SCOPE field in the Logical Block Provisioning log page.
[d] Minimum available LBA count = resource count × $2^{(\text{threshold exponent})}$.
[e] Maximum used LBA count = resource count × $2^{(\text{threshold exponent})}$.

## F.5.5 Operations that occur

Write medium operations occur to LUN 1 that require 2 000 additional logical blocks to be used and write medium operations occur to LUN 2 that require 3 000 additional logical blocks to be used. Used LBA mapping resources on LUN 1 are now 59 000 logical blocks, used LBA mapping resources on LUN 2 are now 106 000 logical blocks, and the combined LBA mapping resources available to both LUN 1 and LUN 2 are 1 073 736 900 logical blocks.

## F.5.6 Shared available, dedicated used, example final log page values

Table F.10 shows the values in the Logical Block Provisioning log page after the operations described in F.5.5 have occurred.

**Table F.10 — Shared available, dedicated used example final log page values**

| LUN | Log page parameter [a] | Resource count [b] | Scope [c] | Description |
|---|---|---|---|---|
| 1 | 0001h | 0007_FFFDh | 10b | The available LBA mapping resource parameter indicates that from 1 073 735 680 logical blocks (i.e. 7_FFFDh threshold sets × 2 048 logical blocks per threshold set) to 1 073 737 727 logical blocks are available for LUN 1 or LUN 2. [d] |
| | 0002h | 0000_001Dh | 01b | The used LBA mapping resource parameter indicates that from 57 345 logical blocks to 59 392 logical blocks (i.e., 1Dh threshold sets × 2 048 logical blocks per threshold set) have been used (i.e., allocated) by LUN 1. [e] |
| 2 | 0001h | 0007_FFFDh | 10b | The available LBA mapping resource parameter indicates that from 1 073 735 680 logical blocks (i.e. 7_FFFDh threshold sets × 2 048 logical blocks per threshold set) to 1 073 737 727 logical blocks are available for LUN 1 or LUN 2. [d] |
| | 0002h | 0000_0034h | 01b | The used LBA mapping resource parameter indicates that from 104 449 logical blocks to 106 496 logical blocks (i.e., 34h threshold sets × 2_048 logical blocks per threshold set) have been used (i.e., allocated) by LUN 2. [e] |

[a] THRESHOLD RESOURCE field in the Logical Block Provisioning mode page (see 6.5.7.2) and the PARAMETER CODE field in Logical Block Provisioning log page (see 6.4.4.2).
[b] RESOURCE COUNT field in the Logical Block Provisioning mode page.
[c] SCOPE field in the Logical Block Provisioning log page.
[d] Minimum available LBA count = resource count × $2^{(\text{threshold exponent})}$.
[e] Maximum used LBA count = resource count × $2^{(\text{threshold exponent})}$.