From: Bob Sheffield, Keith Holt, and Bret Weber – LSI Corporation
To: T10 Committee
Subject: T10/08-044r0 SBC-3 DIF Granularity
Date: 3 January 2008

Related Documents: SBC-3r12

Overview

There has been recent momentum in the storage industry behind the idea of supporting larger logical blocks. This offers the opportunity for improved reliability for systems that use larger logical blocks. Interoperability with existing applications that largely depend on logical blocks having a size of 512 bytes prompt the need for storage virtualization solutions that emulate 512-byte logical blocks for the virtual logical units they expose to applications, but use disk drives formatted with larger logical blocks (e.g., 4096 bytes) as the raw storage for the virtual logical units.

Systems such as these should be able to provide protection information for the virtual logical units exposed to applications. This prompts the need for physical sector formats on the disk drives to accommodate 2^n x (512+8) bytes of information, where 2^n is the virtual device logical block size / disk drive logical block size; in order to provide space for protection information on disk. Furthermore, the disk should be able to check protection information fields even when the granularity of data protected by protection information fields is smaller than the disk blocksize.

This proposal defines the application of protection information so that the logical block size reported by the device server is a power-of-two multiple of the number of bytes in the unit of user data protected by each occurrence of protection information.

In short:

- Define a sub-block as the unit of user data protected by an occurrence of protection information.
- Define the frequency of protected sub-blocks as 2^n such that the logical block size = 2^n x the sub-block size, where $n \in \{0, 1, 2, 3\ldots\}$.
- For type-1 data protection, define the logical block reference tag as containing 2^n x the LBA in the CDB.
- For type-2 data protection, the CDB specifies the value of the first logical block reference tag transferred and the value increments by one for each subsequent sub-block (as opposed to logical block).

This addresses both the need to provide raw block capacity for virtual devices emulating legacy block sizes, and also avoids diluting the effectiveness of the CRC field as logical block sizes increase.

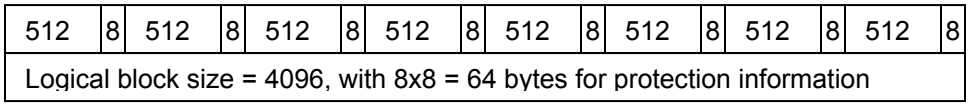

Figure 1 – Multiple protection information fields per logical block

Suggested Changes

3.1.40 protection information: Fields appended to each logical block or sub-block that contain a cyclic redundancy check (CRC), an application tag, and a reference tag. See 4.17.

3.1.xx sub-block: A unit of user data (see 3.1.50) within a logical block (see 3.1.26) that is protected by an instance of protection information (see 3.1.40).

4.4 Logical blocks

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

The first LBA is zero. The last LBA is [n-1], where [n] is the number of logical blocks on the medium accessible by the application client. The READ CAPACITY (10) parameter data (see 5.12.2 and 5.13.2) RETURNED LOGICAL BLOCK ADDRESS field indicates the value of [n-1].

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

a) the application client should enable descriptor format sense data (see SPC-4) in the Control mode page (see SPC-4) and in any REQUEST SENSE commands (see SPC-4) it sends; and
b) the application client should use commands with 8-byte LOGICAL BLOCK ADDRESS fields (e.g., READ CAPACITY (16), READ (16), and WRITE (16)).

NOTE 2 - If a command with a 4-byte LOGICAL BLOCK ADDRESS field accesses logical blocks beyond LBAs FFFFFFFFh and fixed format sense data is used, there is no field in the sense data large enough to report the LBA of an error (see 4.14).

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

The number of bytes of user data contained in a logical block is the logical block length. The parameter data returned by the READ CAPACITY command (see 5.12) describes the logical block length that is used on the medium. The mode parameter block descriptor (see 6.3.2) is used to change the logical block length in direct-access block devices that support changeable logical block lengths. The logical block length ~~should be used to determine~~ does not include the length of protection information and additional information, if any.

The location of a logical block on the medium is not required to have a relationship to the location of any other logical block. However, in a typical direct-access block device, the time to access a logical block at LBA [x+1] after accessing LBA [x] is often less than the time to access some other logical block. The time to access the logical block at LBA [x] and then the logical block at LBA [x+1] need not be less than time to access LBA [x] and then LBA [x+100]. The READ CAPACITY

command issued with a PMI bit set to one may be useful in determining where longer access times occur.

4.17.3 Protection information format

Table 7 defines the ~~placement~~ format of protection information in a ~~logical~~ sub-block.

Table 7 – Sub-block user data and protection information format

Byte/Bit	7	6	5	4	3	2	1	0
0	SUB-BLOCK USER DATA							
n - 1								
n	LOGICAL BLOCK GUARD							
n + 1								
n + 2	LOGICAL BLOCK APPLICATION TAG							
n + 3								
n + 4	LOGICAL BLOCK REFERENCE TAG							
n + 7								

Table 8 defines the placement of protection information within a logical block size of m x n bytes containing m sub-blocks with n-bytes of user data each, where m = 2^x and x is the value of the SUB-BLOCKS_PER_LOGICAL_BLOCK_EXP_REQ field in the FORMAT UNIT parameter list header (see 5.2.2.2).

Table 8 – Placement of protection information in a logical block

Byte/Bit	7	6	5	4	3	2	1	0
0	SUB-BLOCK USER DATA							
n - 1								
n	PROTECTION INFORMATION							
n + 7								
n + 8	SUB-BLOCK USER DATA							
2n+7								
2n + 8	PROTECTION INFORMATION							
2n+15								
…	…							
(m – 1) x (n + 8)	SUB-BLOCK USER DATA							
m x (n + 8) - 9								
m x (n + 8) - 8	PROTECTION INFORMATION							
m x (n + 8) - 1								

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

The LOGICAL BLOCK GUARD field contains the CRC (see 4.17.4) of the contents of the SUB-BLOCK USER DATA field.

The 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 4.17.2.3) or type 2 protection (see 4.17.2.4) is enabled; or
b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF FFFFh, and type 3 protection (see 4.17.2.5) is enabled,

then the device server disables checking of all protection information for the logical block when reading from the medium. Otherwise, the contents of the logical block application tag are not defined by this standard.

The LOGICAL BLOCK APPLICATION TAG field may be modified by a device server if the ATO bit is set to zero in the Control mode page (see SPC-4).

The contents of the LOGICAL BLOCK APPLICATION TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

The LOGICAL BLOCK REFERENCE TAG field of the first sub-block in the first logical block in the data-in buffer and/or data-out buffer shall contain the value specified in table ~~8~~9.

Table ~~8~~9 – Content of the LOGICAL BLOCK REFERENCE TAG field of the first sub-block of the first logical block in the data-in buffer and/or data-out buffer

Protection Type	Content of the LOGICAL BLOCK REFERENCE TAG field of the first sub-block of the first logical block in the data-in buffer and/or data-out buffer
Type-1 protection (see 4.17.2.3)	The least significant four bytes of the quantity m x LBA; where m = 2^x is the number of sub-blocks in a logical block, and LBA is the ~~LBA contained~~ value in the LOGICAL BLOCK ADDRESS field of the command.
Type-2 protection (see 4.17.2.4)	The value in the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field of the command.
Type-3 protection (see 4.17.2.5)	Not defined in this standard.

The LOGICAL BLOCK REFERENCE TAG field subsequent logical blocks in the data-in buffer and/or data-out buffer shall be set as specified in table ~~9~~ 10.

Table ~~9~~10 – Setting of the LOGICAL BLOCK REFERENCE TAG field of the subsequent ~~logical block~~ sub-block in the data-in buffer and/or data-out buffer

Protection Type	The content of the LOGICAL BLOCK REFERENCE TAG field of the sebsequent ~~logical block~~ sub-block in the data-in buffer and/or data-out buffer
Type-1 protection (see 4.17.2.3) and Type-2 protection (see 4.17.2.4)	The logical block reference tag of the previous ~~logical block~~ sub-block plus one.
Type-3 protection (see 4.17.2.5)	Not defined in this standard.

The contents of the LOGICAL BLOCK REFERENCE TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

4.17.4.1 Logical block guard overview

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

Table ~~10~~ 11 defines the CRC polynomials used to generate the logical block guard from the contents of the SUB-BLOCK USER DATA field.

Table ~~10~~**101** – CRC Polynomials

Function	Definition
F(x)	A polynomial representing the transmitted SUB-BLOCK 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 SUB-BLOCK USER DATA field and the coefficient of the lowest order term shall be bit zero of the last byte of the SUB-BLOCK USER DATA field.
F'(x)	A polynomial representing the received SUB-BLOCK USER DATA field.
G(x)	The generator polynomial: $G(x) = x^{16} + x^{15} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (i.e., G(x) = 18BB7h)
R(x)	The remainder polynomial calculated during CRC generation by the transmitter, representing the transmitted LOGICAL BLOCK GUARD field.
R'(x)	A polynomial representing the received LOGICAL BLOCK GUARD field.
RB(x)	The remainder polynomial calculated during CRC checking by the receiver. RB(x) = 0 indicates no error was detected.
RC(x)	The remainder polynomial calculated during CRC checking by the receiver. RC(x) = 0 indicates no error was detected.
QA(x)	The quotient polynomial calculated during CRC generation by the transmitter. The value of QA(x) is not used.
QB(x)	The quotient polynomial calculated during CRC checking by the receiver. The value of QB(x) is not used.
QC(x)	The quotient polynomial calculated during CRC checking by the receiver. The value of QC(x) is not used.
M(x)	A polynomial representing the transmitted SUB-BLOCK USER DATA field followed by the transmitted LOGICAL BLOCK GUARD field.
M'(x)	A polynomial representing the received SUB-BLOCK USER DATA field followed by the received LOGICAL BLOCK GUARD field.

4.17.4.2 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 SUB-BLOCK USER DATA field followed by the LOGICAL BLOCK GUARD field (i.e., F(x) followed by R(x)):

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

4.17.4.3 CRC checking

M'(x) (i.e., the polynomial representing the received SUB-BLOCK USER DATA field followed by the received LOGICAL BLOCK GUARD field) may differ from M(x) (i.e., the polynomial representing the transmitted SUB-BLOCK 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.

4.17.4 CRC test cases

Several CRC test cases are shown in table ~~11~~ 12.

Table ~~11~~102 – 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

4.17.5 Application of protection information

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

1) determine if a logical unit supports protection information using the INQUIRY command (see the PROTECT bit in the standard INQUIRY data in SPC-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 (see the PROT_EN bit in 5.13); and
3) if the logical unit supports protection information and is not formatted to accept protection information, then format the logical unit with protection information enabled.

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

4.17.6 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 ~~12~~ 13 (see 5.1).

Commands that return the length in bytes of each logical block (e.g., the MODE SENSE commands and the READ CAPACITY commands) shall return the length of the combined SUB-BLOCK USER DATA fields and shall not include the length of the protection information (i.e., the LOGICAL BLOCK GUARD field, the LOGICAL BLOCK APPLICATION TAG field, and the LOGICAL BLOCK REFERENCE TAG field) ~~(e.g., if the user data plus the protection information is equal to 520 bytes then 512 is returned)~~ (e.g., if the logical block consists of eight sub-blocks each containing 512 bytes of user data plus the protection information totaling 4160 bytes, then 4096 is returned).

5.2 FORMAT UNIT command

5.2.2 FORMAT UNIT parameter list

5.2.2.2 Parameter list header

The parameter list headers (see table 17 and table 18) provide several optional format control parameters. Device servers that implement these headers provide the application client additional control over the use of the four defect sources, and the format operation. If the application client attempts to select any function not implemented by the device server, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The short parameter list header (see table 17) is used if the LONGLIST bit is set to zero in the FORMAT UNIT CDB.

Table 17 – Short parameter list header

Byte/Bit	7	6	5	4	3	2	1	0
0	SUB-BLOCKS_PER_LOGICAL_BLOCK_EXP_REQ				Reserved	PROTECTION FIELD USAGE		
1	FOV	DPRY	DCRT	STPF	IP	Obsolete	IMMED	Vendor-Specific
2	(MSB)	DEFECT LIST LENGTH						
3								(LSB)

The long parameter list header (see table 18) is used if the LONGLIST bit is set to one in the FORMAT UNIT CDB.

Table 18 – Long parameter list header

Byte/Bit	7	6	5	4	3	2	1	0
0	SUB-BLOCKS_PER_LOGICAL_BLOCK_EXP_REQ				Reserved	PROTECTION FIELD USAGE		
1	FOV	DPRY	DCRT	STPF	IP	Obsolete	IMMED	Vendor-Specific
	Reserved							
	Reserved							
2	(MSB)	DEFECT LIST LENGTH						
3								(LSB)

The SUB-BLOCKS_PER_LOGICAL_BLOCK_EXP_REQ field is not valid and the device server shall ignore the field if the PROTECTION FIELD USAGE field set to 000b. If the PROTECTION FIELD USAGE field is non-zero then the device server shall:

a) format each logical block to contain 2^x sub-blocks, such that each sub-block contains a unit of user data of logical block length / (2^x) bytes and is followed by protection

information for the unit of user data in each sub-block, where x is the value of the SUB-BLOCKS_PER_LOGICAL_BLOCK_EXP_REQ field; or

b) terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST if the logical block length is not divisible by 2^x, or if the device server does not support the number of sub-blocks per logical block requested.

The remainder of the FORMAT UNIT command description is unchanged.

5.3 ORWRITE command

Nothing changes except table footnote h in Table 29 – ORPROTECT field – checking protection information read from the medium (part 3 of 3), and footnote f in Table 30 - ORPROTECT field – checking protection information from the data-out buffer (part 2 of 2), as follows:

h If type 1 protection is enabled, the device server checks the logical block reference tag of each sub-block by comparing it to the lower 4 bytes of a value which is 2^x times the LBA associated with the logical block plus the number of preceding sub-blocks in the logical block, where 2^x is the number of sub-blocks in each logical block. If type 3 protection is enabled, the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. The method for acquiring this knowledge is not defined by this standard.

f If type 1 protection is enabled, the device server checks the logical block reference tag of each sub-block by comparing it to the lower 4 bytes of a value which is 2^x times the LBA associated with the logical block plus the number of preceding sub-blocks in the logical block, where 2^x is the number of sub-blocks in each logical block. If type 3 protection is enabled, the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. The method for acquiring this knowledge is not defined by this standard.

5.7 READ (6) command

Nothing changes except table footnote f in Table 36 – Protection information checking for READ (6), as follows:

f If type 1 protection is enabled, the device server checks the logical block reference tag of each sub-block by comparing it to the lower 4 bytes of a value which is 2^x times the LBA associated with the logical block plus the number of preceding sub-blocks in the logical block, where 2^x is the number of sub-blocks in each logical block. If type 2 protection or type 3 protection is enabled, the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. The method for acquiring this knowledge is not defined by this standard.

5.7 READ (10) command

Nothing changes except table footnote i in Table 38 – RDPROTECT field (part 3 of 3), as follows:

i If type 1 protection is enabled, the device server checks the logical block reference tag of each sub-block by comparing it to the lower 4 bytes of a value which is 2^x times the LBA associated with the logical block plus the number of preceding sub-blocks in the logical block, where 2^x is the number of sub-blocks in each logical block. 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 5.11). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

5.1 READ (32) command

Modify the description of the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field as follows:

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 38 in 5.8), 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 sub block in the first logical block accessed by the command instead of a value based on the LBA (see 4.17.3).

5.13.2 READ CAPACITY (16) parameter data

The READ CAPACITY (16) parameter data is defined in table 46. Any time the READ CAPACITY (16) parameter data changes, the device server should establish a unit attention condition as described in 4.7.

Table 46 – READ CAPACITY (16) parameter data

Byte/Bit	7	6	5	4	3	2	1	0
0	(MSB)							
7				RETURNED LOGICAL BLOCK ADDRESS				(LSB)
8	(MSB)							
11				LOGICAL BLOCK LENGTH IN BYTES				(LSB)
12	~~Reserved~~ SUB-BLOCKS PER LOGICAL BLOCK EXPONENT				P_TYPE			PROT_EN
13	Reserved				LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT			
14	Reserved		(MSB)					
15				LOWEST ALIGNED LOGICAL BLOCK ADDRESS				(LSB)
16								
31				Reserved				

The RETURNED LOGICAL BLOCK ADDRESS field and LOGICAL BLOCK LENGTH IN BYTES field of the READ CAPACITY (16) parameter data are the same as the in the READ CAPACITY (10) parameter data (see 5.12). The maximum value that shall be returned in the RETURNED LOGICAL BLOCK ADDRESS field is FFFFFFFF_FFFFFFFEh.

The protection type (P_TYPE) field and the protection enable (PROT_EN) bit (see table 47) indicate the logical unit's current type of protection.

Table 47 – P_TYPE and PROT_EN bits

PROT_EN	P_TYPE	Description
0	xxxb	The logical unit is formatted to type 0 protection (see 4.17.2.2).
1	000b	The logical unit is formatted to type 1 protection (see 4.17.2.3).
1	001b	The logical unit is formatted to type 2 protection (see 4.17.2.4).
1	010b	The logical unit is formatted to type 3 protection (see 4.17.2.5).
1	011b – 111b	Reserved

The SUB-BLOCKS PER LOGICAL BLOCK EXPONENT field is defined such that there are m sub-blocks within each logical block, where $m = 2^x$, and x is a non-negative integer (i.e., $x \in \{0, 1, 2, 3\ldots\}$) and is the value in the SUB-BLOCKS PER LOGICAL BLOCK EXPONENT field.

Note - nn: x = 0 indicates each a logical block consists of a single sub-block.

The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field is defined in table 49.

Table 48 – 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.	

The LOWEST ALIGNED LOGICAL BLOCK ADDRESS field indicates the LBA of the first logical block that is located the beginning of a physical block (see 4.5).

NOTE 14 - The highest LBA that the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field supports is 3FFFh (i.e., 16 383).

5.22 Verify (10) command

Modify table footnotes in tables as follows:

Table 65 — VRPROTECT field with BYTCHK set to zero - checking protection information read from the medium (part 3 of 3):

a) If type 1 protection is enabled, the device server checks the logical block reference tag of each sub-block by comparing it to the lower 4 bytes of a value which is 2^x times the LBA associated with the logical block plus the number of preceding sub-blocks in the logical block, where 2^x is the number of sub-blocks in each logical block. If type 2 protection or type 3 protection is enabled, the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. 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 5.25). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

Table 66 — VRPROTECT field with BYTCHK set to one - checking protection information read from the medium (part 2 of 2):

b) If type 1 protection is enabled, the device server checks the logical block reference tag of each sub-block by comparing it to the lower 4 bytes of a value which is 2^x times the LBA associated with the logical block plus the number of preceding sub-blocks in the logical block, where 2^x is the number of sub-blocks in each logical block. If type 2 protection or type 3 protection is enabled, the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. 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 5.25). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

Table 67 — VRPROTECT field with BYTCHK set to one - checking protection information from the data-out buffer (part 2 of 2):

f If type 1 protection is enabled, the device server checks the logical block reference tag of each sub-block by comparing it to the lower 4 bytes of a value which is 2^x times the LBA associated with the logical block plus the number of preceding sub-blocks in the logical block, where 2^x is the number of sub-blocks in each logical block. If type 2 protection or type 3 protection is enabled, the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. 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 5.25). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

5.26 WRITE (6) command

The WRITE (6) command (see table 72) requests that the device server transfer the specified logical block(s) from the data-out buffer and write them. Each logical block transferred includes user data but does not include protection information. Each logical block written includes user data and, if the medium is formatted with protection information enabled, protection information.

Table 72 – Write (6) command

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

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 72.

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

See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field.

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

NOTE 20 - For the WRITE (10) command, WRITE (12) command, WRITE (16) command, and WRITE (32) command, a TRANSFER LENGTH field set to zero specifies that no logical blocks are transferred.

The contents of the CONTROL byte are defined in SAM-4.

If a WRITE (6) command is received after protection information is enabled the device server shall set the protection information (see 4.17) as follows as it writes each logical block to the medium:

a) the LOGICAL BLOCK GUARD field set to a properly generated CRC (see 4.17.4);
b) the LOGICAL BLOCK REFERENCE TAG field set to:
   A) the least significant four bytes of the a value that is 2^x times the LBA plus the number of preceding sub-blocks in the logical block, where 2^x is the number of sub-blocks in each logical block, if type 1 protection (see 4.17.2.3) is enabled; or

B) FFFFFFFFh, if type 2 protection (see 4.17.2.4) or type 3 protection (see 4.17.2.5) is enabled;

and

c) the LOGICAL BLOCK APPLICATION TAG field set to:
   A) FFFFh, if the ATO bit is set to one in the Control mode page (see SPC-4); or
   B) any value, if the ATO bit is set to zero in the Control mode page (see SPC-4).

5.27 WRITE (10) command

Table 74 — WRPROTECT field (part 3 of 3)

Modify the indicated table footnotes as follows:

g If the P_TYPE field is set to 000h in the READ CAPACITY (16) parameter data (see 5.13), the device server shall write the least significant four bytes of a value that is 2^x times the ~~each~~ LBA plus the number of preceding sub-blocks in the logical block, where 2^x is the number of sub-blocks in each logical block, into the LOGICAL BLOCK REFERENCE TAG field of each sub-block of the written logical blocks. If the P_TYPE field is not set to 000b, the device server shall write a value of FFFFFFFFh into the LOGICAL BLOCK REFERENCE TAG field of each sub-block of the written logical blocks.

j If type 1 protection is enabled, the device server checks the logical block reference tag by comparing it to the lower 4 bytes of a value that is 2^x times the LBA associated with the logical block plus the number of preceding sub-blocks in the logical block, where 2^x is the number of sub-blocks in each logical block. If type 2 protection or type 3 protection is enabled, the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. 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 5.30). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

5.30 WRITE (32) command

Third paragraph after **Table 78 — WRITE (32) command**

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 74 in 5.27), 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 sub-block in the first logical block accessed by the command instead of a value based on the LBA (see 4.17.3).

5.34 WRITE AND VERIFY (32) command

Third paragraph after **Table 82 — WRITE AND VERIFY (32) command**

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 74 in 5.27), 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 sub-block in the first logical block accessed by the command instead of a value based on the LBA (see 4.17.3).

5.39 WRITE SAME (32) command

Third paragraph after **Table 89 — WRITE SAME (32) command**

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 74 in 5.27), 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 sub-block in the first logical block accessed by the command instead of a value based on the LBA (see 4.17.3).