From: Bob Sheffield – LSI Corporation
To: T10 Committee
Subject: T10/08-044r3 SBC-3 Set Pseudo Format Command
Date: 28 July 2008

Related Documents: SBC-3r15

Revision History:

Revision	Date	Description
r0	1/3/08	Initial version
r1	5/5/08	Clarify justification (4K initiator,…) and add per-command controls
r2	6/10/08	Propose SET PSEUDO FORMAT command
r3	7/28/08	Incorporate recommendations from July 2008 CAP WG: o Elaborate on model section to detail DIF modes using table o Provide examples including physical block alignment o Clarify that SET PSEUDO FORMAT doesn't require reformatting media o Make PFID field two bits with only a value of 01b defined o Clarify the LOGCIAL BLOCK LENGTH IN BYTES field of READ CAPACITY (16) data returns the value pertaining to the specified pseudo logical block format as specified in a prior SET PSEUDO FORMAT command o Consider: a flag in the FORMAT UNIT command to tell the device server the intent is to issue a subsequent SET PSEUDO FORMAT command o Change service action to a MAINTENANCE OUT service action o Clarify that read/write commands specifying a value in the PFID field that has not been established by a prior SET PSEUDO FORMAT command shall return CHECK CONDITION status,… o Define a LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field in the SET PSEUDO FORMAT command to mean to disable the specified pseudo format o Misc. corrections and clarifications

Overview

There has been recent momentum in the storage industry to support larger logical blocks. This offers the opportunity for improved disk reliability and/or other benefits for systems that use larger logical blocks. Interoperability with existing applications that largely depend on logical blocks having a size of 512 bytes prompts 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.

With the move to larger disk block lengths, applications that access data using matching larger block lengths (rather than depending upon emulation) may access data more efficiently by avoiding latencies caused by read-modify-write operations in the emulation layer. So, it stands to reason that both traditional applications that access media using 512-byte blocks and new applications that access media using larger blocks (e.g., 4K-byte) will coexist, perhaps in the same platforms (e.g., virtualized servers).

Likewise, there will be a gradual transition from systems that use 512-byte formatted disks to larger block length media, and the larger block length media may include both media formatted to a larger logical block length as well as media formatted to a larger physical block length but that emulates 512-byte logical blocks.

This leads to a complex intermix of application expected block lengths and disk supported block lengths that, if not appropriately addressed in SCSI standards, leads to a variety of non-interoperability problems, particularly when protection information (PI) is used. The table below lists the relevant combinations, and whether interoperability holds for PI-enabled implementations.

Case	Application client block length	Disk block length		Comment
		Logical	Physical	
1	T[a]	T	T	Traditional usage
2	L[b]	T	T	Not interoperable – mismatched PI field locations and intervals between app and disk.
3	T	L	T	
4	L	L	T	Interoperable, but not defined in SCSI
5	T	T	L	Emulation of traditional block length
6	L	T	L	Not interoperable – mismatched PI field locations and intervals between app and disk.
7	T	L	L	
8	L	L	L	Complete agreement on larger block lengths
[a] **T**: Traditional block length (i.e., 512-bytes) [b] **L**: Large block length (e.g., 4Kbyte – but could be something else like 2K or 8K)				

When protection information is not used, a block virtualization layer may apply a reasonably straightforward mapping to emulate application block lengths that do not match the logical block lengths provided by the disks. This poses an insurmountable challenge, however, when protection information is involved. The reason is that the application expects PI fields to appear at intervals the disk is incapable of providing without violating SBC-3 (i.e., the requirement that PI fields must appear at logical block boundaries).

Cases 1 and 8 are interoperable through traditional usage of SPC-3 where the application logical block length, device logical block length, and the physical block length are all in agreement.

The existing definitions for the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT and LOWEST ALIGNED LOGICAL BLOCK ADDRESS fields in READ CAPACITY (16) parameter data provide interoperability for case 5, albeit with some potential performance impacts due to misalignment of application accesses with physical block boundaries. Interoperability for the remaining 5 cases is not addressed.

This proposal provides for interoperability for cases 2 and 6 listed above by defining compatible DIF usage when the desired application logical block length is larger (by a factor of 2n) than the formatted logical block length of the block storage device. Cases 3 and 7 remain non-interoperable, but these are not cases of general interest. Case 4 is also not of general interest as the trend is to migrate towards larger physical blocks while providing compatibility with smaller logical blocks.

In virtualized server and external storage environments, block storage devices are likely to encounter the need to support an intermix of applications that require both traditional 512-byte access and larger block length access, simultaneously.

Case 6 is of particular interest because it provides a way to align logical blocks for applications that require larger logical blocks with corresponding larger physical blocks, without subverting legacy applications that rely on the 512+8 block-size emulation mode of the block storage device.

A solution to this problem should accommodate future increases in block length, and should provide the framework to establish interoperability with any application, regardless of the block length expected by the application.

The pseudo-format SCSI protocol extensions in this proposal define the framework to allow a disk to provide interoperability with different applications needing different block lengths, at the same time, regardless of the physical and logical block lengths provided by the disk.

This proposal defines a method by which a disk may support, in addition to its native logical block length, one or more pseudo-formats providing larger block lengths for pseudo blocks on media. Applications may specify, on a per command basis, the pseudo-format that matches the application block format required. The SET PSEUDO FORMAT command specifies one or more pseudo-formats to be used to process read and write commands that may (or may not) be supported by the drive. The SET PSEUDO FORMAT command specifies (as a power of two) the number of disk logical blocks that correspond to a pseudo-block, and whether protection information is appended at logical block boundaries or pseudo logical block boundaries.

This document also proposes using two currently reserved bits in the CDBs for read and write type commands, referred to as the pseudo-format ID or PFID field, in the same byte as the GROUP NUMBER field, to specify a pseudo-format to be used for processing that command.

The pseudo format defined in this proposal allows the application of protection information to coincide either with the pseudo logical block boundaries as specified in the CDB, or with the underlying logical block boundaries, depending on the value of the APIPB bit in the SET PSEUDO FORMAT command.

When the APIPB bit is set to zero, protection information fields align with logical block boundaries with the contents set as follows:

- the value of the logical block application tag is not specified in this standard;
- the value in the logical block guard field is the CRC calculated over the user data contained within the logical block;
- for type-1 data protection, the value in the logical block reference tag for each logical block is computed as LBA * 2^n + m + k, truncated to the least significant four bytes, where the LBA is the LBA of the pseudo block containing the logical block, n is the value of the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field established for the specified PFID in a previous SET PSEUDO FORMAT command, m is the offset (i.e., in logical blocks) of the logical block from the logical block that aligns with the beginning of the pseudo logical block, and k is the value of the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field returned in READ CAPACITY (16) data; and
- for type-2 data protection, the value in the logical block reference tag for each logical block is computed as x * 2^n + m, truncated to the least significant four bytes, where the x is the value of the EXPECTED LOGICAL BLOCK REFERENCE TAG field provided in the read/write CDB, n is the value of the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field established for the specified PFID in a previous SET PSEUDO FORMAT command, and m is the offset (in logical blocks) of the logical block from the logical block that aligns with the beginning of the first pseudo logical block transferred by the command.

When the APIPB bit is set to one, protection information fields align with pseudo logical block boundaries with the contents set as follows:

- the value of the logical block application tag is not specified in this standard;
- the value in the logical block guard field is the CRC calculated over the user data contained within the pseudo logical block;
- for type-1 data protection, the value in the logical block reference tag for each pseudo logical block is LBA of the pseudo logical block (i.e., as specified in the CDB) truncated to the least significant four bytes; and
- for type-2 data protection, the value in the logical block reference tag for each pseudo logical block is the value of the EXPECTED LOGICAL BLOCK REFERENCE TAG field + m where m is the offset (i.e., in pseudo logical blocks) of the pseudo logical block from the first pseudo logical block transferred by the command.

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.

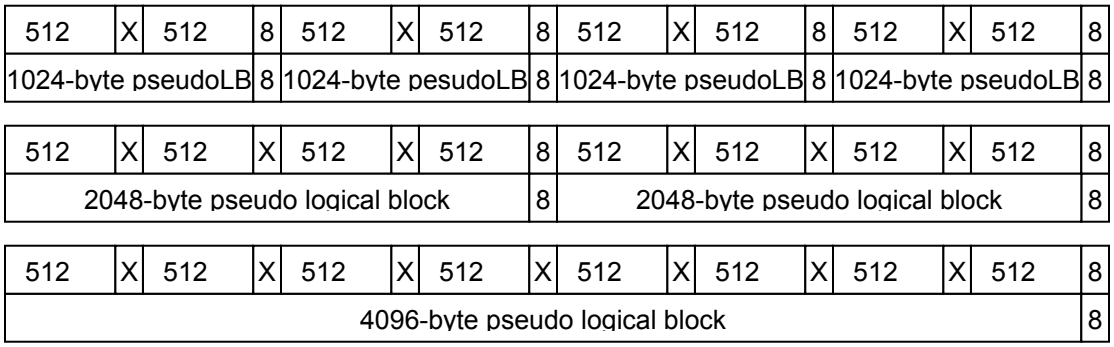

Figure 1 – Example mappings with the APIPB bit set to one

In the above figure, each X represents an instance of protection information that is present on the media, but is neither transferred to or from the application client nor checked by the device server while processing a read or write command with the PFID field set specifying the corresponding pseudo-format. It is possible to read logical blocks without specifying use of a pseudo-format even though those blocks were previously written using a pseudo-format. Protection information fields shown with an X above are written with values that will cause protection information errors if the blocks are subsequently read using a command that specifies a different value in the PFID field.

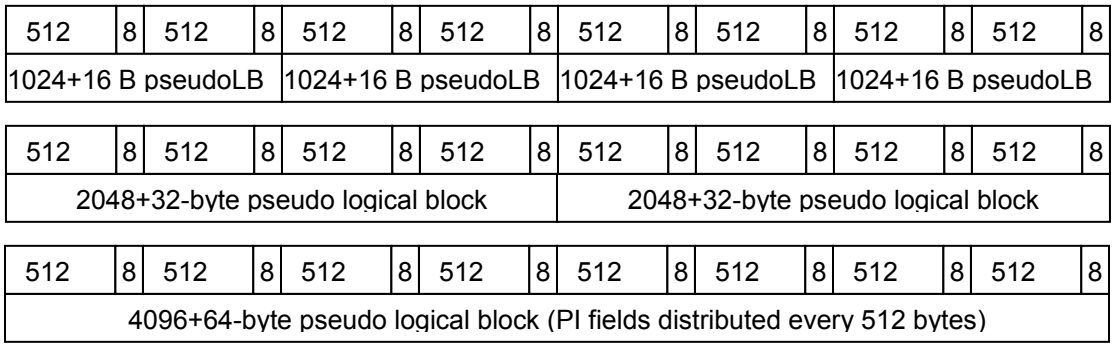

Figure 2 – Example mappings with the APIPB bit set to zero

EDITOR'S NOTE: One requested change from revison-2 of this document that hasn't yet been addressed is the request to represent protection information usage with pseudo blocks as a new type of protection information. The author of this proposal is still working on the appropriate text.

EDITOR'S NOTE: Need to define service action 0Ch of the MAINTENANCE OUT (A4h) command as the SET PSEUDO FORMAT command in SPC-4.

Suggested Changes to SBC-3

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

3.1.xx pseudo logical block: A unit of user data (see 3.1.50) composed of two or more adjacent logical blocks (see 3.1.26) that may be protected by one or more instances of protection information (see 3.1.40).

3.1.xx pseudo logical block address (PLBA): The value used to reference a pseudo logical block (see 3.1.xx).

Add the following to the list of symbols and abbreviations:

PLBA pseudo logical block address (see 3.1.xx)

Modify subclause 4.14.1 Error reporting overview to specify reporting the PLBA instead of the LBA in the INFORMATION *field when applicable.*

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

When a command attempts to access or reference an invalid PLBA, the first invalid PLBA shall be returned in the INFORMATION field of the sense data.

When a recovered read error is reported for a command with the PFID field set to zero, the INFORMATION field of the sense data shall contain the LBA of the last logical block accessed by the command on which a recovered read error occurred.

When a recovered read error is reported for a command with the PFID field set to a non-zero value, the INFORMATION field of the sense data shall contain the PLBA of the last pseudo logical block accessed by the command on which a recovered read error occurred.

When an unrecovered error is reported for a command with the PFID field set to zero, the INFORMATION field of the sense data shall contain the LBA of the logical block on which the unrecovered error occurred.

When an unrecovered error is reported for a command with the PFID field set to a non-zero value, the INFORMATION field of the sense data shall contain the PLBA of the pseudo logical block on which the unrecovered error occurred.

Add subclause 4.6 to describe pseudo logical blocks

4.6 Pseudo logical blocks

An application client may read and write logical blocks using a pseudo logical block address (PLBA) by specifying a non-zero value in the PFID field of the read or write CDB. A PLBA references a pseudo logical block (see 3.1.xx) that is composed of two or more adjacent logical blocks, as specified by the SET PSEUDO FORMAT command for the specified pseudo format ID. The LBA of the first logical block in the pseudo logical block referenced by a PLBA value of zero shall be the LBA of the logical block indicated in the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field returned in the READ CAPACITY (16) parameter data. The LBA of the first logical block in the next pseudo logical block shall be the LBA of the first logical block in the previous pseudo logical block plus 2^n, where n is the value of the LOGICAL BLOCKS PER PSEUDO LOGICAL BLOCK EXPONENT field specified in the SET PEDUDO FORMAT CDB with the same value in the PFID field.

The pattern continues until the number of remaining LBAs, as indicated by the returned LOGICAL BLOCK ADDRESS field in READ CAPACITY (16) parameter data, is less than 2^n.

A non-zero value in the PFID field of a read or write CDB specifies that the application client is specifying a pseudo logical block address in the LOGICAL BLOCK ADDRESS FIELD. If a non-zero value is specified in the PFID field, the field in the CDB specifying the number of logical blocks to process (e.g., the TRANSFER LENGTH field or the NUMBER OF LOGICAL BLOCKS field) shall specify the number of pseudo logical blocks to be processed by the command.

If the device server does not support pseudo block addressing and the PFID field contains a value other than zero, the command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If a read or write command is received with the PFID field set to a value other than zero and the device server has not successfully processed a SET PSEUDO FORMAT command to establish the specified pseudo format, the command shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

The relationship between the LBA of a logical block contained within a pseudo logical block and the PLBA of the pseudo logical block is:

LBA = 2n * PLBA + m + k,

where n is the value of the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field established for the specified PFID in a previous SET PSEUDO FORMAT command, m is the offset (i.e., in logical blocks) of the logical block from the logical block that aligns with the beginning of the pseudo logical block, and k is the value of the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field returned in READ CAPACITY (16) data.

Figure-x shows an example of the relative alignment of logical blocks and the corresponding pseudo logical blocks with the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field set to 3 and the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 7.

LBA = 15	LBA = 16	LBA = 17	LBA = 18	LBA = 19	LBA = 20	LBA = 21	LBA = 22
PLBA = 1							

Figure x – Example of correspondence between LBA and PLBA

Assuming the device server has successfully processed a SET PSEUDO FORMAT command with the PFID field set to 01b, and the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field set to 3; and the READ CAPACITY (16) command returns a value of 7 in the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field, the device server would return the data shown in figure-x in response to either of two READ (12) commands specified as follows:

a) the PFID field set to 00b, the logical block address field set to 15, and the transfer count field set to 8; or
b) the PFID field set to 01b, the logical block address field set to 1, and the transfer count field set to 1.

Protection information may be present either at logical block boundaries or at pseudo logical block boundaries (see 4.17.3).

If the value returned by the device server in the LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field of the READ CAPACITY (16) data is the same value as the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field specified in the SET PSEUDO FORMAT command, then the pseudo logical block boundaries shall align with physical block boundaries.

4.17.3 Protection information format

Table 7 defines the ~~placement~~ format of protection information in a logical block or pseudo logical block.

Table a – user data and protection information format

Byte/Bit	7	6	5	4	3	2	1	0
0	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								

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

The LOGICAL BLOCK GUARD field contains the CRC (see 4.17.4) of the contents of the 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 logical block or pseudo logical block in the data-in buffer and/or data-out buffer shall contain the value specified in table 8.

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

Protection Type	PFID	Content of the LOGICAL BLOCK REFERENCE TAG field of the first logical block or pseudo logical block in the data-in buffer and/or data-out buffer
Type-1 protection (see 4.17.2.3)	= 0	The least significant four bytes of the LBA contained in the LOGICAL BLOCK ADDRESS field of the command.
	≠ 0	See 4.17.4
Type-2 protection (see 4.17.2.4)	= 0	The value in the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field of the command.
	≠ 0	See 4.17.4
Type-3 protection (see 4.17.2.5)	n/a	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.

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

Protection Type	PFID	The content of the LOGICAL BLOCK REFERENCE TAG field of the subsequent logical block or pseudo logical 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)	= 0	The logical block reference tag of the previous logical block plus one.
	≠ 0	See 4.17.4
Type-3 protection (see 4.17.2.5)	n/a	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.

Add subclause 4.17.4 to describe application of protection information with pseudo logical blocks

4.17.4 Protection information with pseudo logical blocks

4.17.4.1 Overview

For commands specifying a non-zero value in the PFID field, the eight bytes of protection information are appended either to each pseudo logical block transferred, or to each logical block within each pseudo logical block, depending on the setting of the APIPB bit for the pseudo format specified in the PFID field. If the device server supports the SET PSEUDO FORMAT command, when processing a FORMAT UNIT command with protection information enabled, the media shall be formatted to accommodate 2^n x 8 bytes of protection information per pseudo logical block, where n is the value of the NUMBER OF LOGICAL BLOCKS IN A PSEUDO LOGICAL BLOCK EXPONENT field specified in the SET PEDUDO FORMAT CDB (i.e., space is provided on media for one instance of protection information for each logical block contained within the pseudo logical block).

Note-x: Providing space for one 8-byte protection information field per logical block is sufficient to meet this requirement since the length of a pseudo logical block is always a multiple of the length of a logical block. Therefore, a FORMAT UNIT command may be processed prior to processing a SET PSEUDO FORMAT command.

4.17.4.2 Protection information alignment with pseudo logical block

Alignment of protection information with pseudo block boundaries is specified using a SET PSEUDO FORMAT command with the APIPB bit set to one for the specified pseudo format. The last protection information field available on media contained within a pseudo logical block shall contain protection information that pertains to the pseudo logical block.

The LOGICAL BLOCK GUARD FIELD shall contain the CRC calculated over all user data contained within the pseudo logical block.

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

Table 10 – Content of the LOGICAL BLOCK REFERENCE TAG field of the first pseudo 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 pseudo 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 PLBA contained 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.

Table 11 – Setting of the LOGICAL BLOCK REFERENCE TAG field of the subsequent pseudo logical block in the data-in buffer and/or data-out buffer

Protection Type	The content of the LOGICAL BLOCK REFERENCE TAG field of the subsequent pseudo logical 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 pseudo logical block plus one.
Type-3 protection (see 4.17.2.5)	Not defined in this standard.

When a command specifying a nonzero value in the PFID field causes data to be written to media, the unused protection information fields for each logical block shall be written as follows:

a) the value stored in the LOGICAL BLOCK GUARD field shall be the one's complement of the CRC calculated over the data in the corresponding logical block,
b) the value stored in the LOGICAL BLOCK REFERENCE TAG field shall be the one's complement of the least significant four bytes of the LBA of the logical block corresponding to that protection information field, and
c) the value stored in the LOGICAL BLOCK APPLICATION TAG field shall be zero.

When a command reads data from media with the PFID field set to a value that is different from the value used to write the data to media (i.e., the read uses a different pseudo block format than the previous write to the same area), if protection information checking is enabled, the device server detects an error in the protection information and reports CHECK condition status with the sense key set to ABORTED COMMAND and the additional sense code set to LOGICAL BLOCK GUARD CHECK FAILED.

4.17.4.3 Protection information alignment with logical block

If the APIPB bit is set to zero for a specified non-zero pseudo format ID, the read or write CDB shall specify data transferred as a range of pseudo logical blocks (i.e., the LOGICAL BLOCK ADDRESS field specifies the starting PLBA and the TRANSFER LENGTH field specifies a count of pseudo logical blocks). However, valid protection information shall be present for each logical block contained within each pseudo block. The contents of the protection information field for each logical block shall be set as follows:

a) the value of the logical block application tag is not specified in this standard;
b) the value in the logical block guard field is the CRC calculated over the user data contained within the logical block;
c) for type-1 data protection, the value in the logical block reference tag for each logical block is computed as PLBA * 2^n + m + k, truncated to the least significant four bytes, where the LBA is the PLBA of the pseudo block containing the logical block, n is the value of the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field established for the specified PFID in a previous SET PSEUDO FORMAT command, m is the offset (i.e., in logical blocks) of the logical block from the logical block that aligns with the beginning of the pseudo logical block, and k is the value of the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field returned in READ CAPACITY (16) data; and
d) for type-2 data protection, the value in the logical block reference tag for each logical block is computed as x * 2^n + m, truncated to the least significant four bytes, where the x is the value of the EXPECTED LOGICAL BLOCK REFERENCE TAG field provided in the read/write CDB, n is the value of the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field established for the specified PFID in a previous SET PSEUDO FORMAT command, and m is the offset (in logical blocks) of the logical block from the logical block that aligns with the beginning of the first pseudo logical block transferred by the command.

4.17.4.4 Protection information and pseudo logical block alignment examples

Figure x shows examples of protection information alignment within pseudo logical blocks when the APIPB bit for the corresponding pseudo format is set to zero.

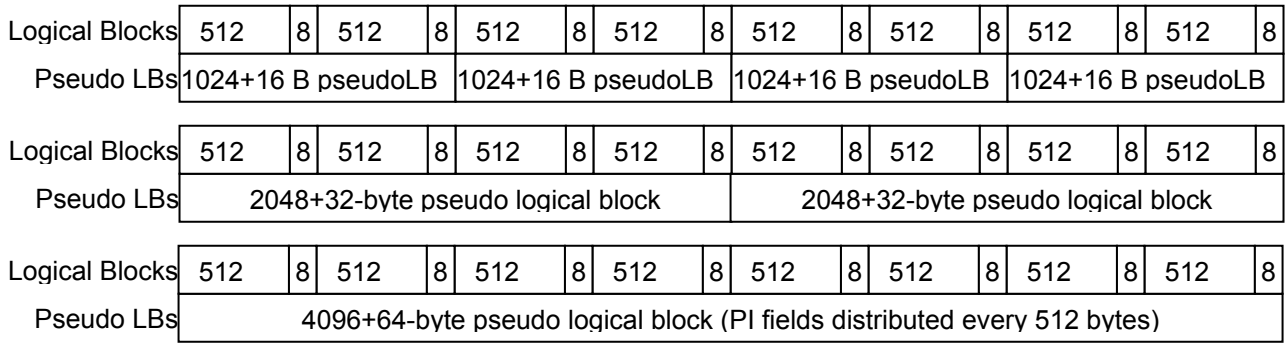

Figure x – Example protection information mappings with the APIPB bit set to zero

Cells shown with a value of 8 indicate 8 bytes of valid protection information transferred and/or checked in the range of data specified by the read or write command specifying use of a pseudo format.

Figure y shows examples of protection information alignment within pseudo logical blocks when the APIPB bit for the corresponding pseudo format is set to one.

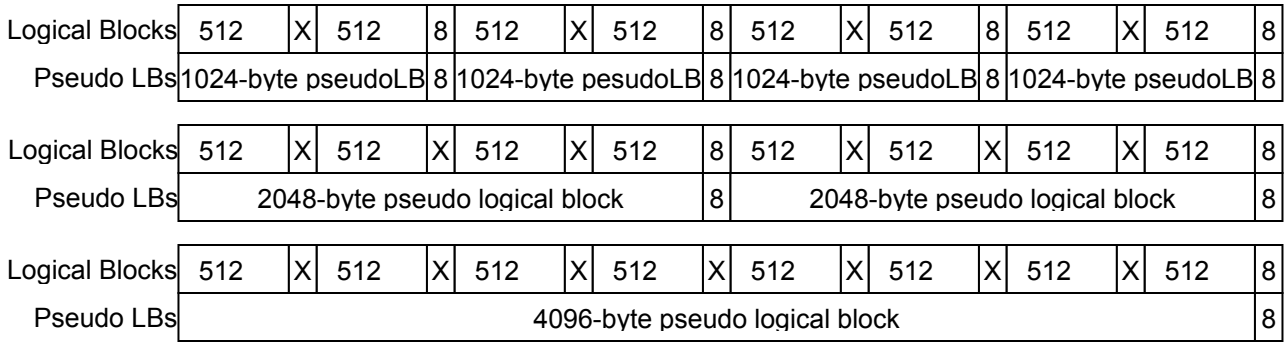

Figure 3 – Example protection information mappings with the APIPB bit set to one

Cells shown with an X indicate represents an instance of protection information that is present on the media, but is neither transferred to or from the application client nor checked by the device server while processing a read or write command with the PFID field set specifying the corresponding pseudo-format. It is possible to read logical blocks without specifying use of a pseudo-format even though those blocks were previously written using a pseudo-format. Protection information fields shown with an X above are written with values that will cause protection information errors if the blocks are subsequently read using a command that specifies a different value in the PFID field (see 4.17.4.2).

Modify the FORMAT UNIT command as follows

5.2 FORMAT UNIT command

5.2.1 FORMAT UNIT command overview

The FORMAT UNIT command (see table 15) 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 6.3.2) 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 (see SPC-4), then that deferred downloaded code shall replace the current operational code.

Any previously established pseudo formats (see 5.x.x) shall be invalidated by processing a FORMAT UNIT command.

The remainder of the FORMAT UNIT command is unchanged

Modify the READ CAPACITY (16) command as follows

5.13 READ CAPACITY (16) command

5.13.1 READ CAPAICTY (16) command overview

The READ CAPACITY (16) command (see table 46) 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 4.17) and is optional otherwise. This command is implemented as a service action of the SERVICE ACTION IN operation code (see A.2). This command may be processed as if it has a HEAD OF QUEUE task attribute (see 4.12).

Table 46 – READ CAPACITY (16) command

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

The OPERATION CODE field and SERVICE ACTION field are defined in SPC-4 and shall be set to the values defined in table 46.

See the READ CAPACITY (10) command (see 5.12) for definitions of the LOGICAL BLOCK ADDRESS field and the PMI bit. If the PFID field is not set to zero, the LOGICAL BLOCK ADDRESS FIELD specifies the PLBA of a pseudo block (see).

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

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

A value of zero in the PFID field requests information returned in the READ CAPACITY (16) parameter data that pertains to the logical block format of the media. A non-zero value in the PFID field requests information returned in the READ CAPACITY (16) parameter data that pertains to the specified pseudo logical block format as specified in a previous SET PSEUDO FORMAT command. The SET PSEUDO FORMAT command (see 5.x.x) defines the encoding of the PFID field.

5.13.2 READ CAPACITY (16) parameter data

The READ CAPACITY (16) parameter data is defined in table 47. 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				P_TYPE			PROT_EN
13	Reserved				LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT			
14	Reserved		(MSB)					
15				LOWEST ALIGNED LOGICAL BLOCK ADDRESS				(LSB)
16	Reserved		PFID		APIPB	LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT		
17~~16~~								
31				Reserved				

If the PFID field in the READ CAPACITY (16) command is set to zero, ~~The~~ 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).

If the PFID field in the READ CAPACITY (16) command is set to a value that corresponds to the PFID of a pseudo format established by a previous SET PSEUDO FORMAT command, then the The returned logical block address field shall return a PLBA and the logical block length field shall be set as follows:

a) If the PMI bit is set to zero, then the device server shall set the RETURNED LOGICAL BLOCK ADDRESS field to the PLBA of the last pseudo logical block on the direct-access block device.
b) If the PMI bit is set to one, then the device server shall set the RETURNED LOGICAL BLOCK ADDRESS field to the last PLBA after that specified in the LOGICAL BLOCK ADDRESS field of the CDB before a substantial vendor-specific delay in data transfer may be encountered.
c) The RETURNED LOGICAL BLOCK ADDRESS shall be greater than or equal to that specified by the LOGICAL BLOCK ADDRESS field in the CDB.
d) The LOGICAL BLOCK LENGTH IN BYTES field contains the number of bytes of user data in the pseudo logical block indicated by the RETURNED LOGICAL BLOCK ADDRESS field. This value does not include protection information or additional information (e.g., ECC bytes) recorded on the medium.

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 48) indicate the logical unit's current type of protection.

Table 48 – P_TYPE field and PROT_EN bit

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

Table 49 – LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field and

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 at 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).

The PFID field shall return the same pseudo format ID value as specified in the PFID field of the READ CAPACITY (16) CDB.

If the PFID field is non-zero, the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field indicates the number of logical blocks contained in each pseudo logical block for the specified pseudo format ID as specified in a previous SET PSEUDO FORMAT command.

If the PFID field is non-zero, the APIPB field indicates whether protection information is included for each logical block or for each pseudo logical block for the specified pseudo format ID as specified in a previous SET PSEUDO FORMAT command.

If the PFID field is zero, the APIPB field and the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field are reserved and shall be set to zero.

Add subclause 5.xx (following 5.18 REASSIGN BLOCKS) describing the SET PSEUDO FORMAT command

5.xx SET PSEUDO FORMAT command

The SET PSEUDO FORMAT command (see table x) specifies use of an alternate method to address logical blocks associated with a pseudo format identifier that may be specified in a read or write command in the PFID field. This command is implemented as a service action of the MAINTENANCE OUT operation code (see A.2). Support for this command is optional.

Table x – SET PSEUDO FORMAT command

Byte/Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (A4h)							
1	Reserved			SERVICE ACTION (0Ch)				
2	Reserved							
3	Reserved	PFID		APIPB	LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT			
4	Reserved							
10								
11	CONTROL							

The OPERATION CODE field and the SERVICE ACTION field are defined in SPC-4 and shall be set to the values defined in table x.

When processing a read or write command that specifies the corresponding pseudo format identifier, a non-zero value in the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field specifies that the device server shall:

a) calculate the LBA of the first logical block to transfer as the product of the LOGICAL BLOCK ADDRESS field of the read or write command times 2^n, where n is the value in the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field plus the value of the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field in the READ CAPACITY (16) parameter data; and
b) calculate the number of logical blocks to transfer as the product of the TRANSFER LENGTH field of the read or write command times 2^n, where n is the value in the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field for the specified pseudo format.

A value of zero in the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field shall cause the device server to disable the pseudo format specified in the PFID field. Any read or write type commands, or READ CAPACITY (16) command received with the PFID field set to the specified value shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. A subsequent SET PSEUDO FORMAT command may establish the same or a new pseudo format associated with the PFID.

The PSEUDO FORMAT IDENTIFIER (PFID) field specifies the pseudo format identifier value the device server shall associate with the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field and APIPB bit values specified in the CDB, and the device server shall apply the pseudo format for the specified pseudo format identifier when that value appears in the PFID field of a read or write command.

Table x defines the encoding of the PFID field.

Table x – PFID field encoding

Code	Description
0	Invalid if specified in the SET PSEUDO FORMAT command. Specifies that a read or write type command or READ CAPACITY (16) command is to process the LOGICAL BLOCK ADDRESS field and TRANSFER LENGTH field in reference to logical block addressing.
1	In a SET PSEUDO FORMAT CDB, establishes a pseudo format for commands specifying a PSID value of 1. For read and write type commands and the READ CAPACITY (16) command, specifies that the LOGICAL BLOCK ADDRESS field and TRANSFER LENGTH field refer to pseudo logical blocks as defined for a PSID value of 1.
2 and 3	Reserved

The ALIGN PROTECTION INFORMATION ON PSEUDO BLOCK (APIPB) bit specifies whether to align protection information, if present, with logical block boundaries or pseudo block boundaries. An APIPB bit set to zero specifies that protection information fields align with logical block boundaries. An APIPB bit set to one specifies that protection information fields align with pseudo logical block boundaries. The APIPB bit and alignment of protection information fields within pseudo logical blocks is described in 4.17.4.

Any pseudo format established by successful processing of this command shall remain active following any reset or power on condition.

Successful processing of a FORMAT UNIT command (see …) shall invalidate all previously established pseudo formats.

Modify the descriptions of the applicable block commands (those that contain a GROUP NUMBER field) as follows

Table x – 10-byte CDB format

Byte/Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	Command dependent							Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved	~~Reserved~~ PFID		GROUP NUMBER				
7	(MSB)	TRANSFER LENGTH (or other length field)						
8								(LSB)
9	CONTROL							

A non-zero value in the PSEUDO FORMAT ID (PFID) field specifies that the LOGICAL BLOCK ADDRESS field contains the PLBA of the first pseudo logical block to be transferred, and that the TRANSFER LENGTH field specifies the number of pseudo logical blocks to transfer, where each pseudo logical block contains 2^n logical blocks, and n is the value in the LOGICAL BLOCKS PER PSEUDO BLOCK EXPONENT field for the specified pseudo format established by the SET PSEUDO FORMAT command defining the pseudo format for the specified pseudo format ID.

Table x – 12-byte CDB format

Byte/Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	Command dependent							Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	(MSB)	TRANSFER LENGTH (or other length field)						
9								(LSB)
10	Reserved	~~Reserved~~ PFID		GROUP NUMBER				
11	CONTROL							

Table x – 16-byte CDB format

Byte/Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	Command dependent							Obsolete
2	(MSB)	LOGICAL BLOCK ADDRESS						
9								(LSB)
10	(MSB)	TRANSFER LENGTH (or other length field)						
13								(LSB)
14	Reserved	~~Reserved~~ PFID		GROUP NUMBER				
15	CONTROL							

Editor's note: Some of the commands show the high-order bit of the VBLE field as "Restricted for MMC-4" because some MMC-4 commands define it as the "streaming" bit. This bit is available to direct-access storage devices because MMC devices (e.g., optical media) have no need to use the streaming bit (which implies tolerance of bit-errors for video applications – not appropriate for direct-access storage devices).

Table x – 32-byte CDB format (example)

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