To: T10 Technical Committee
From: Rob Elliott, HP (elliott@hp.com)
Date: 1 May 2006
Subject: 06-034r3 SBC-3 Physical blocks

Revision history

Revision 0 (4 January 2006) First revision
Revision 1 (3 March 2006) Incorporated comments from January 2006 CAP WG.
Revision 2 (26 April 2006) Incorporated comments from March 2006 CAP WG.
Revision 3 (1 May 2006) Incorporated comments from Gerry Houlder (Seagate).

Related documents

sbc3r04 - SCSI Block Commands - 3 (SBC-3) revision 4
sat-r07 - SCSI to ATA Translation (SAT) revision 7
ANSI INCITS 317-1998 ATA Atachment - 4 with Packet Interface Extension (ATA/ATAPI-4)
T13/d1699r03b - AT Attachment - 8 ATA/ATAPI Command Set (ATA8-ACS) revision 3b

Related web sites

Big Sector consortium - http://www.bigsector.org (with presentation material from Maxtor, Seagate, Western Digital, Hitachi GST, Intel, LSI Logic, and Microsoft)
IDEMA Symposium: HDD Dynamics--Interfaces, Electronics, Architecture and Reliability 6 December 2005: Session 3: HDD Sector Architecture. Slides at:
http://www.idema.org/_smartsite/modules/news/show_news.php?cmd=display&news_id=1230
IDEMA Committee on Long Data Block Standards. Various presentations at:
http://www.idema.org/_smartsite/modules/local/data_file/show_file.php?cmd=standards&cat=103

Overview

ATA devices are starting to support physical sectors that are larger than logical sectors (see the web sites above). For example, the ATA device may use 4096 byte physical sectors but still present 512 byte logical sectors to software. This has several benefits:

a) improved error correction capability (an ECC over a 4096 byte physical sector could tolerate a large burst error, while that same error would be unrecoverable if it occurs within one 512 byte physical sector)
b) increased capacity (one ECC code covering 4096 bytes is shorter than 8 ECC codes covering 512 bytes each)
c) increased data rates (better ECC coverage allows more raw media errors).

The Long Physical Sector feature set and the Long Logical Sector feature set relate to this capability. The SCSI to ATA Translation (SAT) standard defines how to map an ATA device into a SCSI logical unit, but SCSI currently defines no way to report these features. Also, vendors may want to support the same features in native SCSI disk drives.

ATA Long Physical Sector feature set

The Long Physical Sector feature set (ata8acs-r03b section 4.19) allows an ATA device to present a 512 byte logical sector size for ATA media access commands while implementing (behind-the-scenes) a larger physical sector size. The feature set lets the physical sector size be a 2^n multiple of the logical sector size, so 1, 2, 4, 8, ... 32,768 logical blocks correspond to 512, 1024, 2048, 4096, ... 16,777,216 bytes per physical block. IDENTIFY DEVICE data word 106 contains the number of logical sectors per physical sector (2^n where n is 0 through 15).

If a read command accesses less than the physical sector size, the ATA device takes a bit longer to read extra data (probably saving it into a read cache). If a write command accesses less than the physical sector size, the ATA device performs a read-modify-write, which has a noticeable performance impact.

ATA8-ACS also allows logical sectors to not be aligned to the physical sectors (e.g., the ATA device could be designed such that a 1024 byte access at LBA 1 is aligned but LBA 0 is not). IDENTIFY DEVICE data Word 209 defines the offset of LBA 0 within a physical sector (14 bits, to allow up to LBA 16,383). It's not clear why this doesn't support the maximum value supported by word 106 (which would take 15 bits).

This was added because:

a) Master Boot Record (MBR) partitioned disks generally contain a single partition starting at LBA 63. Thus, when using a disk with 4096 byte physical sectors (8 logical sectors per physical sector), performance will be better if LBAs 7, 15, 23, 31, 39, 47, 55, 63, etc. are aligned to the physical sector boundary rather than LBAs 0, 8, 16, 24, 32, 40, 48, 56, 64, etc.
a) EFI partitioned disks generally contain a first partition starting at LBA 34. Performance will be better if LBAs 2, 10, 18, 26, 34, etc. are aligned to the physical sector boundaries rather than LBAs 0, 8, 16, 24, 32, 40, etc.

NOTE 1 - Unified EFI 2.0 recommends that partition tools align partitions to the physical sector boundaries reported by the disk drive. See http://www.uefi.org.

ATA Long Logical Sector feature set

The Long Logical Sector feature set (ata8acs-r03b section 4.20) supports logical sectors that are not 512 bytes. Words 117-118 contain the logical sector size in 16-bit words (with word 106 bit 12 also set to 1).

SCSI already provides the equivalent functionality in the READ CAPACITY data BLOCK LENGTH IN BYTES field and the mode parameter block descriptor BLOCK LENGTH field.

Proposal

This proposal suggests these changes to SBC-3:

a) Add a LOGICAL BLOCKS PER PHYSICAL BLOCK field to the READ CAPACITY (16) data to indicate the number of logical blocks per physical block.
b) Add a LOWEST ALIGNED LOGICAL BLOCK ADDRESS field to the READ CAPACITY (16) data to indicate the lowest LBA that is aligned to a physical block boundary. The field size is the same as the field size of the ATA IDENTIFY DEVICE data's Word 209 (the Offset field) so SAT can fully translate an ATA device.
c) Add a LOGICAL BLOCK ADDRESS TO ALIGN field to FORMAT UNIT to specify an LBA (probably the lowest LBA) that the device server shall align to a physical block boundary (default of 0 means naturally aligned). The field size of 6 bits only supports alignment up to LBA 63, which supports MBR partitioned disks. The need for this feature should dissipate if partition tools start to place partitions on natural boundaries. Software simply writes the starting LBA of its (only or most important) partition to this field (modulo the field size) - it need not perform any calculations based on the LOGICAL BLOCKS PER PHYSICAL BLOCK field.
d) Add a PBLOCK bit to READ LONG and WRITE LONG to access a physical block rather than a logical block. For legacy software compatibility, the default accesses just a logical block. This requires a device server supporting large physical blocks to be able to mark individual logical blocks as containing errors and to use a format for the long data designed that a WRITE LONG to one logical block has no affect on any other logical blocks. If it cannot do so, it must terminate the command.
e) Add a WR_UNCOR bit to WRITE LONG to mark the specified logical or physical block as containing an error. A SATL can directly translate this into the ATA8-ACS WRITE UNCORRECTABLE EXT command, which was added by T13 proposal e02126r6 and induces an error on either a specific physical block (called a "pseudo-uncorrectable error") or a specific logical block (called a "flagged error"). ATA8-ACS does not define a WRITE LONG command, so it is not possible to translate the existing SCSI WRITE LONG semantics into anything guaranteed to have the same effect in ATA.

NOTE 2 - This feature is particularly important for copy utilities that need to replicate bad logical blocks on disks, or RAID controllers that need to mark an error on a failed regenerated logical block. If legacy software is using SCSI WRITE LONG to do this, however, SAT cannot determine whether it is trying to cause an error or not. It would help if ATA resurrected its WRITE LONG command (last defined in ATA/ATAPI-4) and added a WRITE LONG EXT to map directly to the SCSI versions. Otherwise, SCSI software will have to change to use this new bit.

f) Allow FORMAT UNIT to mark more than the LBA(s) specified in the defect list as defective. The device server may choose to mark all the logical blocks sharing the same physical block as defective during the format operation. This is not allowed during REASSIGN BLOCKS or WRITE LONG, however, since side-effects from those commands would break software compatibility. Since the entire disk is being formatted at once with FORMAT UNIT, it doesn't matter for FORMAT UNIT.

Suggested changes to chapter 3 (Definitions)

3.1.1 direct-access block device: A device that is capable of containing data stored in logical blocks that each have a unique logical block address. See 4.1.

3.1.2 logical block: A set of data bytes accessed and referenced as a unit by the application client. See 4.4.

3.1.3 logical block address (LBA): The value used to reference a logical block (see 4.4).

3.1.4 logical block length: The number of bytes of user data in a logical block (see 4.4).

3.1.5 physical block: A set of data bytes accessed as a unit by the device server. See 4.x.

3.1.6 physical block length: The number of bytes of user data in a physical block (see 4.x).

Suggested changes to chapter 4 (Model)

4.1 Direct-access block device type model overview

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

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

Direct-access block devices store data for later retrieval in logical blocks. Logical blocks contain user data, may contain protection information accessible to the application client, and may contain additional information not normally accessible to the application client (e.g., an ECC). The number of bytes of user data contained in each logical block is the logical block length. The logical block length is greater than or equal to one byte and should be even. Most direct-access block devices support a logical block length of 512 bytes and some support additional logical block lengths (e.g., 520 or 4096 bytes). The logical block length does not include the length of protection information and additional information, if any, that are associated with the logical block. The logical block length is the same for all logical blocks on the medium.

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

Logical blocks are stored by a process that causes localized changes or transitions within a medium. The changes made to the medium to store the logical blocks may be volatile (i.e., not retained through power cycles) or non-volatile (i.e., retained through power cycles). The medium may contain vendor specific information that is not addressable through an LBA. Such data may include defect management data and other device management information.

Editor's Note 1: Much of the material in the preceding 3 paragraphs is also stated in 4.4. The two sections should be compared and duplicate material removed.

4.2 Media examples

...

4.2.2 Rotating media

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

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

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

A rotating media-based direct-access block device is ready when the disks are rotating at the correct speed and the read-write circuitry is powered and ready to access the data, and may require a START STOP UNIT command (see 5.17) to bring the logical unit to the ready state.

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

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

4.2.3 Memory media

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

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

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

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

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

4.3 Removable medium

...

4.4 Logical blocks

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

The first ~~logical block address~~LBA is zero. The last ~~logical block address~~LBA is [n-1], where [n] is the number of logical blocks on the medium accessible by the application client. ~~A~~The READ CAPACITY ~~command should be used to determine~~ data (see 5.10.2 and 5.11.2) RETURNED LOGICAL BLOCK ADDRESS field indicates the value of [n-1].

~~Logical block address~~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-3) in the Control mode page (see SPC-3) and in any REQUEST SENSE commands (see SPC-3) it sends; and
b) the application client should use commands with 8-byte LOGICAL BLOCK ADDRESS fields (e.g., READ CAPACITY (16), READ (16), and WRITE (16)).

NOTE 3 - If a command with a 4-byte LOGICAL BLOCK ADDRESS field accesses logical blocks beyond ~~logical block address~~LBA FFFFFFFFh and fixed format sense data is used, there is no field in the sense data large enough to report the ~~logical block address~~LBA of an error (see 4.13).

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

The number of bytes of user data contained in a logical block is the logical block length. The parameter data returned by the READ CAPACITY command (see 5.10) 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 does not include the length of protection information and additional information, if any.

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

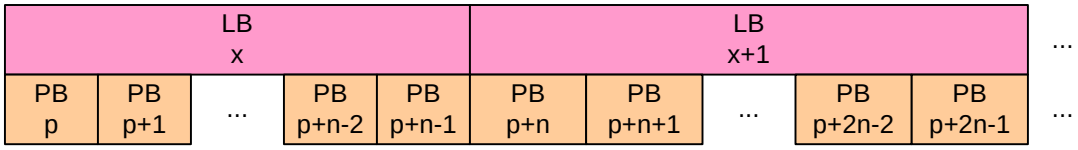
A physical block is a set of data bytes on the medium accessed by the device server as a unit. A physical block may contain:

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 4 096 bytes with a logical block length of 512 bytes);
b) a 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 512 bytes with a logical block length of 4 096 bytes).

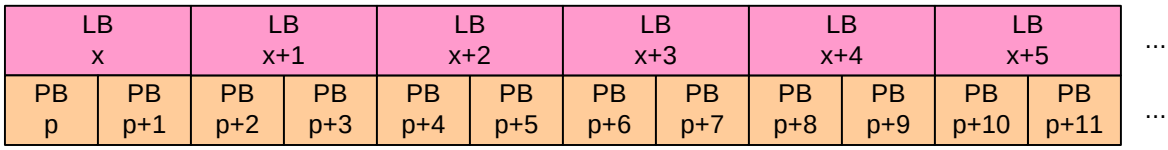
Each physical block includes additional information not normally accessible to the application client (e.g., an ECC) that the device server uses to manage storage and retrieval. The device server shall be able to mark individual logical blocks as containing uncorrectable errors.

Figure 1 shows examples of logical blocks and physical blocks.

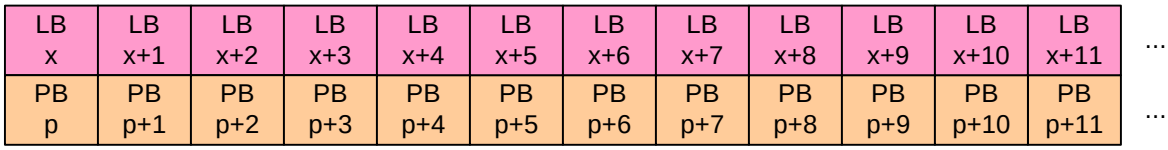
n physical blocks per logical block:



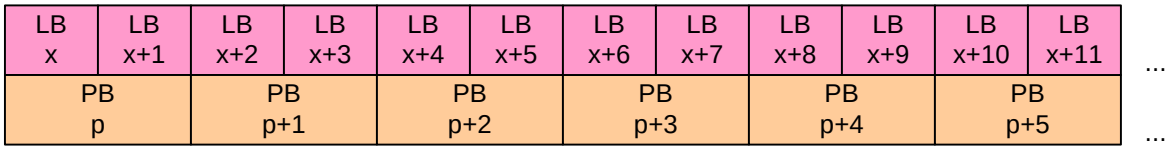
2 physical blocks per logical block:



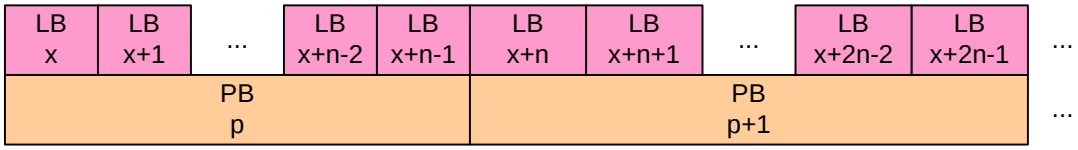
1 logical block per physical block:



2 logical blocks per physical block:



n logical blocks per physical block:



Key:
LB n = logical block n with LBA n
PB n = physical block n

Figure 1 — Logical blocks and physical blocks examples

When there are n logical blocks per physical block, only one of n logical blocks are aligned to the physical block boundaries. To avoid incurring a performance penalty on media access commands, application clients should:

a) specify a logical block that is aligned to a physical block boundary (e.g., set the LOGICAL BLOCK ADDRESS field to an integral multiple of the LBA indicated by the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field in the READ CAPACITY (16) parameter data (see 5.11.2)); and
b) access an integral number of physical blocks (e.g., set the TRANSFER LENGTH field to an integral multiple of the number of logical blocks indicated by the LOGICAL BLOCKS PER PHYSICAL BLOCK field in the READ CAPACITY (16) parameter data (see 5.11.2)).

Figure 2 shows examples of logical blocks with different alignments to physical blocks.

LOGICAL BLOCKS PER PHYSICAL BLOCK field set to 1h (indicating 2^1 logical blocks per physical block), LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 0:

...	LB x-2	LB x-1	LB x	LB x+1	LB x+2	LB x+3	LB x+4	LB x+5	LB x+6	LB x+7	LB x+8	LB x+9	...
...	PB p		PB p+1		PB p+2		PB p+3		PB p+4		PB p+5		...

LOGICAL BLOCKS PER PHYSICAL BLOCK field set to 1h (indicating 2^1 logical blocks per physical block), LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 1:

...	*LB x-1*	LB x	LB x+1	LB x+2	LB x+3	LB x+4	LB x+5	LB x+6	LB x+7	LB x+8	LB x+9	LB x+10	...
...	PB p		PB p+1		PB p+2		PB p+3		PB p+4		PB p+5		...

LOGICAL BLOCKS PER PHYSICAL BLOCK field set to 2h (indicating 2^2 logical blocks per physical block), LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 0:

...	*LB x-4*	*LB x-3*	*LB x-2*	*LB x-1*	LB x	LB x+1	LB x+2	LB x+3	LB x+4	LB x+5	LB x+6	LB x+7	...
...	PB p				PB p+1				PB p+3				...

LOGICAL BLOCKS PER PHYSICAL BLOCK field set to 2h (indicating 2^2 logical blocks per physical block), LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 1:

...	*LB x-3*	*LB x-2*	*LB x-1*	LB x	LB x+1	LB x+2	LB x+3	LB x+4	LB x+5	LB x+6	LB x+7	LB x+8	...
...	PB p				PB p+1				PB p+3				...

LOGICAL BLOCKS PER PHYSICAL BLOCK field set to 2h (indicating 2^2 logical blocks per physical block), LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 2:

...	*LB x-2*	*LB x-1*	LB x	LB x+1	LB x+2	LB x+3	LB x+4	LB x+5	LB x+6	LB x+7	LB x+8	LB x+9	...
...	PB p				PB p+1				PB p+3				...

LOGICAL BLOCKS PER PHYSICAL BLOCK field set to 2h (indicating 2^2 logical blocks per physical block), LOWEST ALIGNED LOGICAL BLOCK ADDRESS field set to 3:

...	*LB x-1*	LB x	LB x+1	LB x+2	LB x+3	LB x+4	LB x+5	LB x+6	LB x+7	LB x+8	LB x+9	LB x+10	...
...	PB p				PB p+1				PB p+3				...

Key:
LB n = logical block n with LBA n
PB n = physical block n
The LOGICAL BLOCKS PER PHYSICAL BLOCK field and LOWEST ALIGNED LOGICAL BLOCK ADDRESS field are in the READ CAPACITY (16) data.

Note:
If LB x is the first logical block (i.e., LBA 0), then the logical blocks labeled x-1, x-2, etc. do not exist.
If LB x is the last logical block, then the logical blocks labeled x+1, x+2, etc. do not exist.

Figure 2 — Logical block to physical block alignment examples

4.6 Initialization

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

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

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

NOTE 4 - Mode parameter block descriptors read with the MODE SENSE command before a FORMAT UNIT completes return information that may not reflect the true state of the medium.

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

Any time the parameter data returned by the READ CAPACITY (10) command (see 5.10) or the READ CAPACITY (16) command (see 5.11) changes (e.g., when a FORMAT UNIT command or a MODE SELECT command completes changing the number of logical blocks, logical block ~~size~~length, protection information, or reference tag ownership values, or when a vendor-specific mechanism causes a change), the device server should establish a unit attention condition for the initiator port associated with each I_T nexus except the I_T nexus on which the command causing the change was received with an additional sense code of CAPACITY DATA HAS CHANGED.

NOTE 5 - Logical units compliant with previous versions of this standard did not establish a unit attention condition.

4.8 Medium defects

Any medium has the potential for defects that cause data to be lost. Therefore, each logical block may contain additional information that allows the detection of changes to the user data and protection information, if any, caused by defects in the medium or other phenomena, and may also allow the data to be reconstructed following the detection of such a change (e.g., ECC bytes). Some direct-access block devices allow the application client to examine and modify the additional information by using the READ LONG commands and the WRITE LONG commands (see 5.14, 5.15, 5.33, and 5.34). The application client may use the WRITE LONG commands to induce a defect to test the defect detection logic of the direct-access block device or to emulate an unrecoverable logical block when generating a mirror copy.

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

a) Primary defect list (PLIST). This is the list of defects, which may be supplied by the original manufacturer of the device or medium, that are considered permanent defects. The PLIST is located outside of the application client accessible logical block space. The PLIST is accessible by the device server for reference during the format operation, but it is not accessible by the application client except through the READ DEFECT DATA commands (see 5.10 and 5.13). Once created, the original PLIST shall not change;
b) Logical unit certification list (CLIST). This list includes defects detected by the device server during an optional certification process performed during the FORMAT UNIT command. This list shall be added to the GLIST;
c) Data defect list (DLIST). This list of defects may be supplied by the application client to the device server during the FORMAT UNIT command. This list shall be added to the GLIST; and

d) Grown defect list (GLIST). The GLIST includes all defects sent by the application client (i.e., the DLIST) or detected by the device server (i.e., the CLIST). The GLIST does not include the PLIST. If the CMPLST bit is set to zero, the GLIST shall include DLISTs provided to the device server during the previous and the current FORMAT UNIT commands. The GLIST shall also include:
   A) defects detected by the format operation during medium certification;
   B) defects previously identified with a REASSIGN BLOCKS command (see 5.16); and
   C) defects previously detected by the device server and automatically reallocated.

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

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

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

Suggested changes to chapter 5 (Commands)

5.2 FORMAT UNIT command

5.2.1 FORMAT UNIT command overview

The FORMAT UNIT command (see table 14) 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-3). In addition, the device server may certify the medium and create control structures for the management of the medium and defects. The degree that the medium is altered by this command is vendor-specific.

If a device server receives a FORMAT UNIT command before receiving a MODE SELECT command with a mode parameter block descriptor, the device server shall use the number of 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.

Table 14 — FORMAT UNIT command

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

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

of this command increase the application client's control over defect management. The application client may specify:

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

While performing a format operation, the device server shall respond to commands attempting to enter into the task set except INQUIRY commands, REPORT LUNS commands, and REQUEST SENSE commands with CHECK CONDITION status with the sense key set to NOT READY and the additional sense code set to LOGICAL UNIT NOT READY, FORMAT IN PROGRESS. Handling of commands already in the task set is vendor-specific.

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

A format protection information (FMTPINFO) bit (see table 19) specifies if the device server enables or disables the use of protection information.

The reference tag own request (RTO_REQ) bit (see table 19) specifies whether the application client or the

device server has ownership of the LOGICAL BLOCK REFERENCE TAG field in protection information (see 4.16.3).

Following a successful format, the P_TYPE field in the READ CAPACITY (16) parameter data (see 5.11.1) indicates the type of protection currently in effect on the logical unit.

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

A LONGLIST bit set to zero specifies that the parameter list, if any, contains a short parameter list header as defined in table 17. A LONGLIST bit set to one specifies that the parameter list, if any, contains a long parameter list header as defined in table 18. If the FMTDATA bit is set to zero, the LONGLIST bit shall be ignored.

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

A FMTDATA bit set to one specifies that the FORMAT UNIT parameter list (see table 16) shall be transferred from the data-out buffer. The parameter list consists of a parameter list header, followed by an optional initialization pattern descriptor, followed by an optional defect list.

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

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

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

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

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

The DEFECT LIST FORMAT field specifies the format of the address descriptors in the defect list if the FMTDATA bit is set to one (see table 15).

Table 15 defines the address descriptor usage for the FORMAT UNIT command.

Table 15 — FORMAT UNIT command address descriptor usage

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

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

5.2.2 FORMAT UNIT parameter list

5.2.2.1 FORMAT UNIT parameter list overview

Table 16 defines the FORMAT UNIT parameter list.

Table 16 — 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 17 or table 18 in 5.2.2.2)							
	Initialization pattern descriptor (if any)(see table 20 in 5.2.2.3)							
	Defect list (if any)							

The parameter list header is defined in 5.2.2.2.

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

The defect list, if any, contains address descriptors (see 5.2.2.4) each specifying a location on the medium that the device server shall exclude from the application client accessible part. This is called the DLIST(see 4.8). The device server shall maintain the current logical block to physical block alignment (see 4.5) for logical blocks not specified in the defect 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	Reserved					PROTECTION FIELDS 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	Reserved					PROTECTION FIELDS USAGE		
1	FOV	DPRY	DCRT	STPF	IP	Obsolete	IMMED	Vendor specific
2	Reserved							
3	Reserved		LOGICAL BLOCK ADDRESS TO ALIGN					
4	(MSB)		DEFECT LIST LENGTH					
7								(LSB)

The PROTECTION FIELD USAGE field in combination with the FMTPINFO bit and the RTO_REQ bit (see table 19) specifies the requested protection type (see 4.16.2).

Table 19 — FMTPINFO bit, RTO_REQ bit, and PROTECTION FIELD USAGE field

					Description
1	1				

[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 command (see 5.11.1) 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 (e.g., if the logical block length is 512, then the formatted logical block length is 520). Following a successful format, the PROT_EN bit in the READ CAPACITY (16) parameter data (see 5.11.1) indicates whether protection information (see 4.16) is enabled.

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

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

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

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

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

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

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

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

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

NOTE 6 - The use of the FMTDATA bit, the CMPLST bit, and the parameter list header allow the application client to control the source of the defect lists used by the FORMAT UNIT command. Setting the DEFECT LIST LENGTH field to zero allows the application client to control the use of PLIST and CLIST without having to specify a DLIST.

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

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

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

The LOGICAL BLOCK ADDRESS TO ALIGN field specifies the LBA of a logical block that the device server should locate at the beginning of a physical block. Each logical block with an LBA that is an integral multiple of this LBA should be located at the beginning of a physical block.

NOTE 7 - The device server reports the lowest LBA it was able to align in the LOWEST ALIGNED LOGICAL BLOCK ADDRESS field in the READ CAPACITY (16) data (see 5.11.2).

Short block format address descriptors and long block format address descriptors should be in ascending order. Bytes from index format address descriptors and physical sector format address descriptors shall be in ascending order. More than one physical or logical block may be affected by each address descriptor. If the address descriptors are not in the required order, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

5.2.2.3 Initialization pattern descriptor

The initialization pattern descriptor specifies that the device server initialize logical blocks to a specified pattern. The initialization pattern descriptor (see table 20) is sent to the device server as part of the FORMAT UNIT parameter list.

Table 20 — Initialization pattern descriptor

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

The initialization pattern modifier (IP MODIFIER) field (see table 21) specifies the type and location of a header that modifies the initialization pattern.

Table 21 — Initialization pattern modifier (IP MODIFIER) field

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

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

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

When the SI bit is set to one, the device server need not write the initialization pattern over the header and other header and other parts of the medium not previously accessible to the application client. If the device server is unable to write over any part of the medium that is currently accessible to the application client or may be made accessible to the application client in the future (e.g., by clearing the defect list), it shall terminate the command with CHECK CONDITION status with the sense key set to MEDIUM ERROR and the

additional sense code set to the appropriate value for the condition. The device server shall attempt to rewrite all remaining parts of the medium even if some parts are not able to be rewritten.

The INITIALIZATION PATTERN TYPE field (see table 22) specifies the type of pattern the device server shall use to initialize each logical block within the application client accessible part of the medium. All bytes within a logical block shall be written with the initialization pattern. The initialization pattern is modified by the IP MODIFIER field as described in table 21.

Table 22 — INITIALIZATION PATTERN TYPE **field**

Code	Description
00h	Use a default initialization pattern [a]
01h	Repeat the pattern specified in the INITIALIZATION PATTERN field as required to fill the logical block [b]
02h - 7Fh	Reserved
80h - FFh	Vendor-specific

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

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

The INITIALIZATION PATTERN LENGTH field specifies the number of bytes contained in the INITIALIZATION PATTERN field. If the initialization pattern length exceeds the current logical block length the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

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

5.2.2.4 Address descriptor formats

5.2.2.4.1 Address descriptor formats overview

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

The format type of an address descriptor is specified with:

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

Table 23 defines the types of address descriptors.

Table 23 — Address descriptor formats

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

5.2.2.4.2 Short block format address descriptor

A format type of 000b specifies the short block format address descriptor defined in table 24.

Table 24 — Short block format address descriptor (000b)

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

For the FORMAT UNIT command, the SHORT BLOCK ADDRESS field contains the four-byte LBA of a defect. The device server may consider logical blocks in addition to the one specified by this descriptor as containing defects (e.g., if multiple logical blocks are contained within a physical block).

For the READ DEFECT DATA commands, the SHORT BLOCK ADDRESS field contains a vendor-specific four-byte value.

For the Translate Address diagnostic pages, the SHORT BLOCK ADDRESS field contains a four-byte LBA or a vendor-specific four-byte value that is greater than the capacity of the medium.

Editor's Note 2: Above used to be 1 paragraph; propose breaking it into 3 paragraphs as shown

---

5.2.2.4.3 Long block format address descriptor

A format type of 011b specifies the long block format address descriptor defined in table 25.

Table 25 — Long block format address descriptor (011b)

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

For the FORMAT UNIT command, the LONG BLOCK ADDRESS field contains the eight-byte ~~logical block address~~LBA of a defect. The device server may consider logical blocks in addition to the one specified by this descriptor as containing defects (e.g., if multiple logical blocks are contained within a physical block).

For the READ DEFECT DATA commands, the LONG BLOCK ADDRESS field contains a vendor-specific eight-byte value.

For the Translate Address diagnostic pages, the LONG BLOCK ADDRESS field contains a ~~four~~eight-byte LBA or a vendor-specific ~~four~~ eight-byte value that is greater than the capacity of the medium.

Editor's Note 3: Above used to be 1 paragraph; propose breaking it into 3 paragraphs as shown

5.2.2.4.4 Bytes from index format address descriptor

A format type of 100b specifies the bytes from index address descriptor defined in table 26. For the FORMAT UNIT command and the READ DEFECT DATA commands, this descriptor specifies the location of a defect that is either the length of one track or is no more than eight bytes long. For the Translate Address diagnostic pages, this descriptor specifies the location of a track or the first byte or last byte of an area.

Table 26 — Bytes from index format address descriptor (100b)

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

The CYLINDER NUMBER field contains the cylinder number.

The HEAD NUMBER field contains the head number.

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

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

5.2.2.4.5 Physical sector format address descriptor

A format type of 101b specifies the physical sector address descriptor defined in table 27. For the FORMAT UNIT command and the READ DEFECT DATA commands, this descriptor specifies the location of a defect that is either the length of one track or the length of one sector. For the Translate Address diagnostic pages, this descriptor specifies the location of a track or a sector.

Table 27 — Physical sector format address descriptor (101b)

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

The CYLINDER NUMBER field contains the cylinder number.

The HEAD NUMBER field contains the head number.

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

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

5.5 READ (6) command

...

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

...

5.10 READ CAPACITY (10) command

...

5.10.2 READ CAPACITY (10) parameter data

The READ CAPACITY (10) parameter data is defined in table 39. Any time the READ CAPACITY (10) parameter data changes, the device server should establish a unit attention condition as described in 4.6.

Table 28 — READ CAPACITY (10) parameter data

Byte\Bit	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)

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

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

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

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

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

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

The 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. This value does not include protection information or additional information (e.g., ECC bytes) recorded on the medium.

5.11 READ CAPACITY (16) command

...

5.11.2 READ CAPACITY (16) parameter data

The READ CAPACITY (16) parameter data is defined in table 29. Any time the READ CAPACITY (16) parameter data changes, the device server should establish a unit attention condition as described in 4.6.

Table 29 — 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			RTO_EN	PROT_EN
13			Reserved			LOGICAL BLOCKS PER PHYSICAL BLOCK		
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 5.10). The maximum value that shall be returned in the RETURNED LOGICAL BLOCK ADDRESS field is FFFFFFFF_FFFFFFFEh.

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

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

The LOGICAL BLOCKS PER PHYSICAL BLOCK field is defined in table 30. If the LOGICAL BLOCKS PER PHYSICAL BLOCK field is set to a non-zero value and the device server supports a WRITE LONG command (see 5.33 and 5.34), it shall support the WR_UNCOR bit and the PBLOCK bit in the WRITE LONG CDB. If the LOGICAL BLOCKS PER PHYSICAL BLOCK field is set to a non-zero value and the device server supports a READ LONG command (see 5.14 and 5.15), it shall support the PBLOCK bit in the READ LONG CDB.

Table 30 — LOGICAL BLOCKS PER PHYSICAL BLOCK field

Code	Description
0	2^0 (i.e., 1) logical block per physical block (see 4.x), or more than one physical block per logical block
n > 0	2^n logical blocks per physical block

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.x). Each logical block with an LBA that is an integral multiple of this LBA is located at the beginning of a physical block.

5.3 PRE-FETCH

...

The LOGICAL BLOCK ADDRESS field specifies the first logical block accessed by this command. If the ~~logical block address~~specified LBA 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.

Editor's Note 4: above paragraph included to show that the standard LOGICAL BLOCK ADDRESS field definition is not 100% suitable for READ LONG/WRITE LONG ("first" is misleading)

5.14 READ LONG (10) command

The READ LONG (10) command (see table 31) requests that the device server transfer data from a single logical 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 physical block is being transferred:
   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);~~.~~

   and

b) if a logical block is being transferred:
   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.

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

Table 31 — 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)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Reserved							
7	(MSB)							
8	BYTE TRANSFER LENGTH							(LSB)
9	CONTROL							

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

The LOGICAL BLOCK ADDRESS field specifies a logical block. If the specified LBA 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.

If the additional information contain an ECC, any other additional bytes that are correctable by ECC should be included (e.g., a data synchronization mark within the area covered by ECC). It is not required for the ECC bytes to be at the end of the user data or protection information, if any; however, the ECC bytes should be in the same order as they are on the medium.

A physical block (PBLOCK) bit set to one specifies that the device server shall return the entire physical block containing the specified logical block. A PBLOCK bit set to zero specifies that the device server shall return bytes representing only the specified logical block.

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

The BYTE TRANSFER LENGTH field specifies the number of bytes of data that shall be read from the specified logical 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, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. In the sense data (see 4.13 and SPC-3), the VALID and ILI bits shall each be set to one and the INFORMATION field shall be set to the difference (i.e., residue) of the requested byte transfer length minus the actual available data length in bytes. Negative values shall be indicated by two's complement notation.

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

5.15 READ LONG (16) command

The READ LONG (16) command (see table 32) requests that the device server transfer data from a single logical 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 physical block is being transferred:
   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).

   and

b) if a logical block is being transferred:
   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.

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

processing this command. This command is implemented as a service action of the SERVICE ACTION IN operation code (see A.2).

Table 32 — READ LONG (16) command

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

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

5.16 REASSIGN BLOCKS command

5.16.1 REASSIGN BLOCKS command overview

The REASSIGN BLOCKS command (see table 33) requests that the device server reassign defective logical blocks to another area on the medium set aside for this purpose. The device server should also record the location of the defective logical blocks in the GLIST, if supported. This command shall not alter the contents of the PLIST (see 4.8).

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

NOTE 11 - The effect of specifying a logical block to be reassigned that previously has been reassigned is to reassign the logical block again. Although not likely, over the life of the medium, a logical block may be assigned to multiple physical block addresses until no more spare locations remain on the medium.

Table 33 — REASSIGN BLOCKS command

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

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

5.16.2 REASSIGN BLOCKS parameter list

The REASSIGN BLOCKS parameter list (see table 34) contains a four-byte parameter list header followed by a defective LBA list containing one or more LBAs.

Table 34 — REASSIGN BLOCKS parameter list

Byte\Bit	7	6	5	4	3	2	1	0
0	Parameter list header (see table 35 or table 36)							
3								
4	DEFECTIVE LBA LIST (if any)							
n								

If LONGLIST is set to zero, the parameter list header is defined in table 35.

Table 35 — REASSIGN BLOCKS short parameter list header

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

If LONGLIST is set to one, the parameter list header is defined in table 36.

Table 36 — REASSIGN BLOCKS long parameter list header

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

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

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

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

If the direct-access block device has insufficient capacity to reassign all of the specified logical blocks, the device server shall terminate the command with CHECK CONDITION status with the sense key set to HARDWARE ERROR and the additional sense code set to NO DEFECT SPARE LOCATION AVAILABLE.

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

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

NOTE 12 - If the REASSIGN BLOCKS command returns CHECK CONDITION status and the sense data COMMAND-SPECIFIC INFORMATION field contains a valid LBA, the application client should remove all LBAs from the defective LBA list prior to the one returned in the COMMAND-SPECIFIC INFORMATION field. If the sense key is MEDIUM ERROR and the INFORMATION field contains the valid LBA, the application client should insert that new defective LBA into the defective LBA list and reissue the REASSIGN BLOCKS command with the new defective LBA list. Otherwise, the application client should perform any corrective action indicated by the sense data and then reissue the REASSIGN BLOCKS command with the new defective LBA list.

5.17 START STOP UNIT command

The START STOP UNIT command (see table 53) requests that the device server change the power condition of the logical unit (see 4.15) or load or eject the medium. This includes specifying that the device server enable or disable the direct-access block device for medium access operations by controlling power conditions and timers.

Logical units that contain cache shall write all cached logical blocks to the medium (e.g., as they would do in response to a SYNCHRONIZE CACHE command (see 5.18 and 5.19) with the SYNC_NV bit set to zero, the LOGICAL BLOCK ADDRESS field set to zero, and the NUMBER OF LOGICAL BLOCKS field set to zero) prior to entering into any power condition that prevents accessing the medium (e.g., before the rotating media spindle motor is stopped during transition to the stopped power condition).

...

5.18 SYNCHRONIZE CACHE (10) command

...

Table 37 — SYNCHRONIZE CACHE (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0 - 6	...							
7	(MSB)	NUMBER OF LOGICAL BLOCKS						
8								(LSB)
9	CONTROL							

...

The NUMBER OF LOGICAL BLOCKS field specifies the number of logical blocks that shall be synchronized, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A NUMBER OF LOGICAL BLOCKS field set to zero specifies that all logical blocks starting with the one specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium shall be synchronized. If the logical block address plus the number of logical blocks exceeds the capacity of the medium, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

A logical block within the range that is not in cache is not considered an error.

5.19 SYNCHRONIZE CACHE (16) command

.....

Table 38 — SYNCHRONIZE CACHE (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0 - 9	...							
10	(MSB)	NUMBER OF LOGICAL BLOCKS						
13								(LSB)
14 - 15	...							

...

5.20 VERIFY (10) command

...

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

...

5.33 WRITE LONG (10) command

The WRITE LONG (10) command (see table 39) requests that the device server mark a logical or physical block as containing an error, or transfer data for a single logical or physical block from the data-out buffer and write it to the medium. The data written shall be the same length and shall be in the same order as the data returned by the READ LONG (10) command (see). The device server shall write the logical or physical block to the medium, and shall not return GOOD status until the logical block has actually been written on the medium.

Table 39 — 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)	BYTE TRANSFER LENGTH						
8								(LSB)
9	CONTROL							

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

A correction disabled (COR_DIS) bit set to zero specifies that, when the specified logical block is read, the device server shall perform normal error recovery on that logical block. A COR_DIS bit set to one specifies specifies that, when the specified logical block is read, the device server shall:

a) perform no error recovery on that logical block including any read error recovery enabled by the Read-Write Error Recovery mode page (see 6.3.5);

b) perform no automatic reallocation of that logical block including any automatic reallocation enabled by the Read-Write Error Recovery mode page;
c) not consider errors on logical blocks to be informational exception conditions as defined in the Information Exceptions Control mode page (see SPC-4)~~.~~; and
d) return CHECK CONDITION status with the sense key set to MEDIUM ERROR and the additional sense code set to READ ERROR - LBA MARKED BAD BY APPLICATION CLIENT.

The condition established by the COR_DIS bit being set to one shall remain in effect until the logical block is written by any means (e.g., any WRITE command, WRITE SAME command, FORMAT command, or another WRITE LONG command specifying the same logical block with the COR_DIS bit set to zero).

The write uncorrectable error (WR_UNCOR) bit and physical block (PBLOCK) bit are defined in table 40.

Table 40 — WR_UNCOR bit and PBLOCK bit

WR_UNCOR	PBLOCK	Description
0	0	Write the specified logical block
0	1	Write the physical block containing the specified logical block
1	0	Mark the specified logical block as containing an uncorrectable error, ignore the BYTE TRANSFER LENGTH field, and shall transfer no data
1	1	Mark the physical block containing the specified logical block as containing an uncorrectable error, ignore the BYTE TRANSFER LENGTH field, and shall transfer no data

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

5.34 WRITE LONG (16) command

The WRITE LONG (16) command (see table 41) requests that the device server mark a logical or physical block as containing an error, or transfer data for a single logical or physical block from the data-out buffer and write it to the medium. The data written shall be the same length and shall be in the same order as the data returned by the READ LONG (16) command (see). The device server shall write the logical or physical block to the medium, and shall not return GOOD status until the logical block has actually been written on the medium. This command is implemented as a service action of the SERVICE ACTION OUT operation code (see A.2).

Table 41 — 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		Reserved						
11								
12	(MSB)	BYTE TRANSFER LENGTH						
13								(LSB)
14	Reserved							
15	CONTROL							

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

5.35 WRITE SAME (10) command

...

Table 79 — WRITE SAME (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0 - 6	...							
7	(MSB)	NUMBER OF LOGICAL BLOCKS						
8								(LSB)
9	...							

Table 80 describes the LBDATA bit and the PBDATA bit.

Table 80 — LBDATA bit and PBDATA bit

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

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

The NUMBER OF LOGICAL BLOCKS field specifies the number of contiguous logical blocks to be written, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A NUMBER OF LOGICAL BLOCKS field set to zero specifies that the device server write all the logical blocks starting with the one specified in the LOGICAL BLOCK ADDRESS field to the last logical block on the medium. If the logical block address plus the number of logical blocks exceeds the capacity of the medium, the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

5.36 WRITE SAME (16) command

...

Table 79 — WRITE SAME (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0 - 9	...							
10	(MSB)	NUMBER OF LOGICAL BLOCKS						
13								(LSB)
14 - 15	...							

5.36 WRITE SAME (32) command

...

Table 79 — WRITE SAME (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0 - 27	...							
28	(MSB)	NUMBER OF LOGICAL BLOCKS						
31								(LSB)

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

Suggested changes to chapter 6 (Parameters)

6.3.2 Mode parameter block descriptors

6.3.2.1 Mode parameter block descriptors overview

If the device server returns a mode parameter block descriptor, it shall return a short LBA mode parameter block descriptor (see 6.3.2.2) 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 the device server returns a mode parameter block descriptor and the number of logical blocks is greater than FFFFFFFFh, it may return a long LBA mode parameter block descriptor (see 6.3.2.3) in the mode parameter data in response to a MODE SENSE (10) command with the LLBAA bit set to one.

If the application client sends a mode parameter block descriptor in the mode parameter list, it shall send a short LBA mode parameter block descriptor (see 6.3.2.2) for a MODE SELECT (6) command.

If the application client sends a mode parameter block descriptor in the mode parameter list, it may send a long LBA mode parameter block descriptor (see 6.3.2.3) 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 of MODE PARAMETERS CHANGED (see SPC-3 and SAM-3) when the block descriptor values are changed.

6.3.2.2 Short LBA mode parameter block descriptor

Table 112 defines the 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-3).

Table 112 — 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-3) 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 the READ CAPACITY command (see 5.11) rather than the MODE SELECT command.

On 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.

On 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, the device server shall return a value of FFFFFFFFh.

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-3), 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, 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, 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, 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, the logical unit shall not become format corrupt. 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 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 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 FFFFFFFFh, then the MODE SELECT 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 PARAMETER LIST. The logical unit shall retain its previous block descriptor settings; or
d) If the NUMBER OF LOGICAL BLOCKS field is set to FFFFFFFFh, 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, the logical unit shall not become format corrupt. 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 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 this capacity setting shall take effect when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command).

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-3) 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 the READ CAPACITY command (see 5.11) rather than the MODE SELECT command.

6.3.2.3 Long LBA mode parameter block descriptor

Table 113 defines the 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-3).

Table 113 — 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-3) 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 the READ CAPACITY command (see 5.11) rather than the MODE SELECT command.

On 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-3), 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, 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, 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, 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, the logical unit shall not become format corrupt. 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 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 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 FFFFFFFF FFFFFFFFh, 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 FFFFFFFF FFFFFFFFh, 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, the logical unit shall not become format corrupt. 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 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 this capacity setting shall take effect when the new logical block length takes effect (i.e., after a successful FORMAT UNIT command).

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-3) 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 the READ CAPACITY command (see 5.11) rather than the MODE SELECT command.

6.3.3 Caching mode page

...

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 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.