To: INCITS Technical Committee T10
From: Fred Knight, Network Appliance
Email: knight@netapp.com
Date: Nov 17, 2008
Subject: SBC-3 Thin Provisioning Commands

1) Revision history

Revision 0 (July 7, 2008) First revision (r0)

Revision 1 (Aug 22, 2008) Revision 1

Split the data path from the management path (create a new proposal for management path). Remove all pools and pool management constructs. This proposal now covers the data path only. Also use the already existing ATA TRIM terminology.

Revision 2 (Sept 5, 2008) Revision 2

Move text from the command clause to the model clause and clarify some language.

Revision 3 (Oct 2, 2008) Revision 3

Introduce LBA state machine with 2 states (hole and allocated) and descriptions of state changes. Remove references to application client and retaining or discarding data, and replace with LBA state changes. Change TRIM back to PUNCH

Revision 4 (Nov 4, 2008) Revision 4

Convert to mapped/unmapped terminology, convert state machine to standard format, redefine read after unmap (previously trim) to match ATA, add PR conflict table entry, CbCS permission table entry, clarify impact of Write Protect on UNMAP command (it fails if write protected), add TPRZ bit (indicating unmapped LBAs read as zero), clarify VERIFY command operation on unmapped LBA, add field to specify max number of extents per UNMAP command to the blocks VPD page, and account for XCOPY (SPC).

Revision 5 (Nov 17, 2008)

Create unmap operation used by UNMAP command, create lists from several long sentences, propose specific ASC/Q values, update the PRE-FETCH command, add the REASSIGN BLOCKS command behavior for unmapped LBAs, clarify the VERIFY command.

2) Related documents

spc4r16 – SCSI Primary Commands – 4
sbc3r15 – SCSI Block Commands – 3
ssc3r04a – SCSI Sequential Commands – 3
08-341r0 – Thin Provisioning Management Commands
T13/e07154r6 – ATA8-ACS2 TRIM proposal (accepted)
T13/e08137 – ATA8-ACS2 DRAT – Deterministic Read After Trim (proposal).

3) Overview

Traditional storage devices pre-allocate physical storage for every possible logical block. There is a fixed one-to-one relationship between the physical storage and the logical storage (every logical block is permanently mapped to a physical block). Generally speaking, the physical capacity of the device is always the same as the logical capacity of the device (plus spares if any). The READ CAPACITY command reports the usable number of logical blocks to the application client. Historically, this has been referred to simply as the capacity of the device. These devices are fully provisioned.

Thinly provisioned devices also report the capacity in the READ CAPACITY command, but they do not allocate (or map) their physical storage in the same way that fully provisioned devices do. Thinly provisioned devices do not necessarily have a permanent one-to-one relationship between the physical storage and the logical storage. Thinly provisioned devices may report a different capacity (in the READ CAPACITY command), than their actual physical capacity. These devices often report a larger capacity than the actual physical capacity for storing user data.

One typical use of storage is a creation and deletion process. Files are created, possibly modified, and saved as new files (with the old one being deleted). Databases are created, where records are added, updated, and deleted.

Fully provisioned storage must allocate space to retain all possible data represented by every block described by the logical capacity (their physical capacity must be the same (or greater) than their reported logical capacity). These devices are always capable of receiving write data into the pre-defined and pre-allocated space.

Thinly provisioned devices may or may not pre-allocate space to retain write data. When a write is received, physical storage may be allocated from a pool of available storage to retain the write data and a mapping established between the location of that physical storage and the appropriate location within the logical capacity (a physical to LBA mapping). As long as this allocation and mapping process is successful, the write operates in the same way that it does on a fully provisioned storage device. However, if all the available physical capacity has been used, and no space can be allocated to retain the write data, the write operation must fail. This failure must have a new unique ASCQ.

In addition, to aid application clients it is desired to notify the client before it actually reaches the point when the failure occurs. These may return a UNIT ATTENTION if the I/O needs to be retried to succeed. These are new types of status conditions to return to the application client, and as such need new ASCQ values to define these conditions. This is needed as part of the I/O path so that error recovery can be synchronized. Out of band techniques would not enable the needed synchronization for error recovery. For example, the application may be involved in notification of a storage administrator to take corrective action, or explicitly taking corrective action of its own. To synchronize that action with the I/O requires this be part of the I/O path.

Event	Sense Key	NEW ASC/Qs	ASC/Q
Temporary lack of physical blocks – write not done, retry required	NOT READY	SPACE ALLOCATION IN PROCESS	04/14
Persistent lack of physical blocks – write not done, retry will not help	DATA PROTECT	SPACE ALLOCATION FAILED – WRITE PROTECT	27/07
Soft Threshold crossed – write not done, retry required	UNIT ATTENTION	PROVISIONING SOFT THRESHOLD REACHED (*)	00/1F 38/10 3F/15 0F/*
(*) – a unit attention condition is established for the initiator port associated with every I_T nexus with the ASC/Q set to PROVISIONING SOFT THRESHOLD REACHED.			

Note: ASC/Q values above are suggestions to start discussion; the final values are to be assigned by the editor.

When the host no longer needs to retain the data (such as when a host file is deleted, or a database record is deleted), there is no specific action required by a fully provisioned device. However, a thinly provisioned device may benefit by knowing about this event and be able to return the physical blocks containing this "deleted" data to a pool of available blocks. Since the data has been deleted, the storage device need not retain the contents of those blocks. If those LBAs are accessed by an application client (a READ is done), the storage device would be free to return any data particular data (zeros, -1, etc). This "delete" function is done via the UNMAP command.

Other possible use cases exist for individual disks, SSD devices, backplane RAID controllers and external RAID controllers.

This proposal defines commands and error codes for the operation of thinly provisioned devices. The UNMAP command includes a method to supply a list of extent descriptors (LBA and length) to a device server.

Management functions will be presented in a separate proposal.

Existing text is shown in **BLACK**, new text is shown in **RED**, and comments (not to be included) are shown in **BLUE**.

Proposal:

3.1.22 format corrupt: a vendor-specific condition in which the application client may not be able to perform read operations, write operations, unmap operations, or verify operations. See 4.7.

<…>

3.1.32 logical unit reset: A condition resulting from the events defined by SAM-4 in which the logical unit performs the logical unit reset operations described in SAM-4, this standard, and other applicable command standards (see table 13 in 5.1).

3.1.32a mapped: A state of an LBA in which there exists a known relationship to a physical block.

3.1.33 media: Plural of medium.

<…>

3.1.54 unit attention condition: A state that a logical unit (see 3.1.30) maintains while the logical unit has asynchronous status information to report to the initiator ports associated with one or more I_T nexuses (see 3.1.25). See SAM-4.

3.1.54a unmapped: A state of an LBA in which the relationship to a physical block is not defined.

3.1.55 unrecovered error: An error for which a device server is unable to read or write a logical block within the recovery limits specified in the Read-Write Error Recovery mode page (see 6.3.5) and the Verify Error Recovery mode page (see 6.3.6).

4 Direct-access block device type 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-4, SPC-4, 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 LBA, which is either four bytes (i.e., a short LBA) or eight bytes (i.e., a long LBA) in length. The LBAs on a 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. An unmap operation causes a change in the relationship between LBAs and physical blocks and may cause a change in the data returned by a subsequent read operation.

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.

<...>

4.4 Logical Blocks

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

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

LBAs are no larger than 8 bytes. Some commands support only 4-byte (i.e., short) LOGICAL BLOCK ADDRESS fields (e.g., READ CAPACITY (10), READ (10), and WRITE (10)). If the capacity exceeds that accessible with short LBAs, then the device server returns a capacity of FFFF_FFFFh in response to a READ CAPACITY (10) command, indicating that:

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

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

If a command is received that references or attempts to access a logical block not within the capacity of the medium, then the device server terminates the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code

set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The device server may terminate the command 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 device server in response to a READ CAPACITY command (see 5.12) describes the logical block length that is used on the medium. The mode parameter block descriptor (see 6.3.2) is used by an application client to change the logical block length in direct-access block devices that support changeable logical block lengths. The logical block length should be used to determine does not include the length of protection information and additional information, if any.

The location of a logical block on the medium is not required to have a relationship to the location of any other logical block. However, in a typical direct-access block device, the time to access a logical block at LBA [x+1] after accessing LBA [x] is often less than the time to access some other logical block. The time to access the logical block at LBA [x] and then the logical block at LBA [x+1] need not be less than time to access LBA [x] and then LBA [x+100]. The READ CAPACITY command issued with a PMI bit set to one may be useful in determining where longer access times occur.

4.4.1 Logical Block Provisioning

4.4.1.1 Provisioning Overview

Each LBA may be mapped or unmapped. For LBAs that are mapped, there is a known relationship to a physical block. For LBAs that are unmapped, the relationship between an LBA and a physical block is not defined. Figure A shows the relationships of LBAs to physical blocks in these two states.

LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7
PB	PB	UNMAPPED	PB	UNMAPPED	UNMAPPED	PB	PB

LBA 0	LBA 1	LBA 2	LBA 3	LBA 4	LBA 5	LBA 6	LBA 7
PB		UNMAPPED		PB		UNMAPPED	

Key:
LBA n = logical block with LBA n
PB = a unique physical block
UNMAPPED = the relationship to a physical block is not defined

Figure A – Mapped and Unmapped Logical Blocks

Transitions between the mapped and unmapped state are shown in figure b.

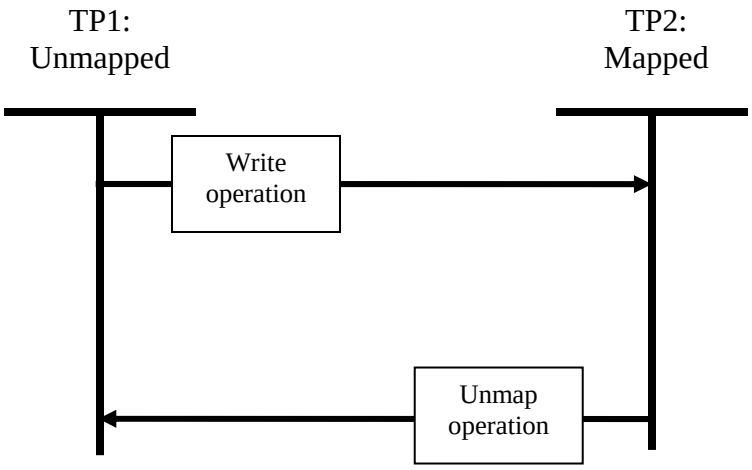

Figure B – Logical Block State Transitions

4.4.1.2 TP2: Mapped state

When in the mapped state, a relationship exists between the LBA and a physical block. User data and protection information read without error from a logical block in the mapped state shall be the user data and protection information that was most recently written to that LBA. If data has not been written to that LBA, user data and protection information read from the LBA shall be the user data and protection information that was established during initialization (see 4.7).

4.4.1.2.1 Transition TP2:Mapped state to TP1:Unmapped state

This transition occurs when an unmap operation completes without error (see 4.4.1.5.2). As a result of this transition, the user data retrieved by the next read operation or verify operation is indeterminate.

4.4.1.3 TP1: Unmapped state

When in the unmapped state, no relationship is defined between the LBA and a physical block (see figure A). For an LBA in the unmapped state, user data and protection information retrieved during a read operation or verify operation:

a) shall not be retrieved from data that was previously written to any other LBA; and
b) shall not change until:
   A. a subsequent write operation to that LBA completes without error (i.e., all read operations or verify operations to an LBA shall ~~return~~ retrieve the same data until a subsequent write operation to that LBA completes without error);
   ~~B. a read operation to that LBA completes with CHECK CONDITION status with the sense key set to MEDIUM ERROR or HARDWARE ERROR; or~~<replaced with change to 5.18 – REASSIGN BLOCK Command>
   ~~C. a power on occurs;~~

If protection information is enabled, the device server shall use a default value of FFFF_FFFF_FFFF_FFFFh as the protection information for logical blocks that are in the unmapped state.

If the TPRZ bit is set to one in the READ CAPACITY (16) parameter data (see 5.13.2), then for an LBA in the unmapped state, the user data retrieved during a read operation or verify operation shall have all bits set to zero.

4.4.1.3.1 Transition TP1:Unmapped state to TP2:Mapped state

This transition:

a) shall occur when a write operation completes without error. As part of processing a write operation, the device server may cause LBAs other than those specified by that write operation to transition from unmapped state to mapped state; or
b) may occur at any other time for vendor specific reasons.

For each LBA that transitions from the unmapped state to the mapped state without transferring data for the logical block from the data-out buffer, the contents of the logical block after the transition shall be the same as if a read operation of the unmapped LBA had been followed by a write operation to that LBA using the data retrieved by that read operation.

4.4.1.4 Full provisioning

A fully provisioned logical unit ensures that a sufficient number of physical blocks are available to retain user data for all logical blocks within the logical unit's reported capacity (see 5.12 - READ CAPACITY (10) and 5.13 - READ CAPACITY(16)). Fully provisioned logical units provide a known relationship between all logical blocks and physical blocks (see figures 2 and 3).

The initial state of all logical blocks on a fully provisioned logical unit is mapped. Logical Blocks on fully provisioned logical units shall not transition out of the mapped state.

Support for full provisioning is indicated by the TPE bit in the READ CAPACITY (16) parameter data (if supported) set to zero.

4.4.1.5 Thin Provisioning

4.4.1.5.1 Thin Provisioning Overview

A thin provisioned logical unit may report a capacity (see 5.12 - READ CAPACITY (10) and 5.13 - READ CAPACITY (16)) larger than the number of mapped LBAs. A thin provisioned logical unit may or may not have sufficient physical blocks to retain user data transferred as a result of a write operation and storing the write data on the medium.

The initial state of each LBA on a thin provisioned logical unit is unmapped (see figure B).

A device server that supports thin provisioning may also support one or more soft threshold values. The method of setting the soft threshold values is outside the scope of this standard.

Support for thin provisioning is indicated by the TPE bit in the READ CAPACITY (16) PARAMETER DATA. Logical units implementing thin provisioning shall:

a) support the READ CAPACITY (16) command;
b) set the TPE bit to one in the READ CAPACITY(16) parameter data (see 6.4.3);
c) support the UNMAP command (see 5.x.1); and
d) set the MAXIMUM UNMAP PARAMETER LIST LENGTH field in the BLOCK LIMITS VPD page (see 6.4.3) to a value greater than or equal to one.

4.4.1.5.2 UNMAP Operation

An unmap operation transitions one or more LBAs to the unmapped state. More than one physical block may be affected by an unmapped operation. The data in all other mapped logical blocks on the medium shall be preserved.

4.4.1.5.3 Thin Provisioning and Protection Information

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

a) mapped state shall be as described in 4.17; and
b) unmapped state shall be as described in 4.4.1.3.

4.4.1.5.4 Resource Exhaustion Considerations

If a write operation is received and a temporary lack of physical block resources prevents the logical unit from storing the write data on the medium, then the device server shall terminate the command with the sense key set to NOT READY and the additional sense code set to SPACE ALLOCATION IN PROCESS. The recommended application client recovery action is to issue the command again at a later time.

If a write operation is received and a persistent lack of physical block resources prevents the logical unit from storing the write data on the medium, then the device server shall terminate the command with the sense key set to DATA PROTECT and the additional sense code set to SPACE ALLOCATION FAILED. Application client recovery actions for this status are outside the scope of this standard.

If a write operation is received that causes the number of available logical block resources to drop below a soft threshold value and the TPNER bit (see 6.3.5) is set to zero, then the device server shall terminate the command and establish a unit attention condition for the initiator port associated with every I_T nexus with the additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED. The recommended application client recovery action is to issue the command again. Further recovery actions (e.g., administrator notification) are outside the scope of this standard.

4.5 Physical blocks

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

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

Each physical block includes additional information not normally accessible to the application client (e.g., an ECC) that the device server uses to manage storage and retrieval.

If the device server supports the COR_DIS bit and/or the WR_UNCOR bit in a WRITE LONG command (see 5.35 and 5.36), then the device server shall have the capability of marking

individual logical blocks as containing pseudo uncorrectable errors with correction enabled (see 3.1.45) or with correction disabled (see 3.1.46).

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

Figure 2 shows examples of logical blocks and physical blocks, where LBA 0 is aligned to a physical block boundary.

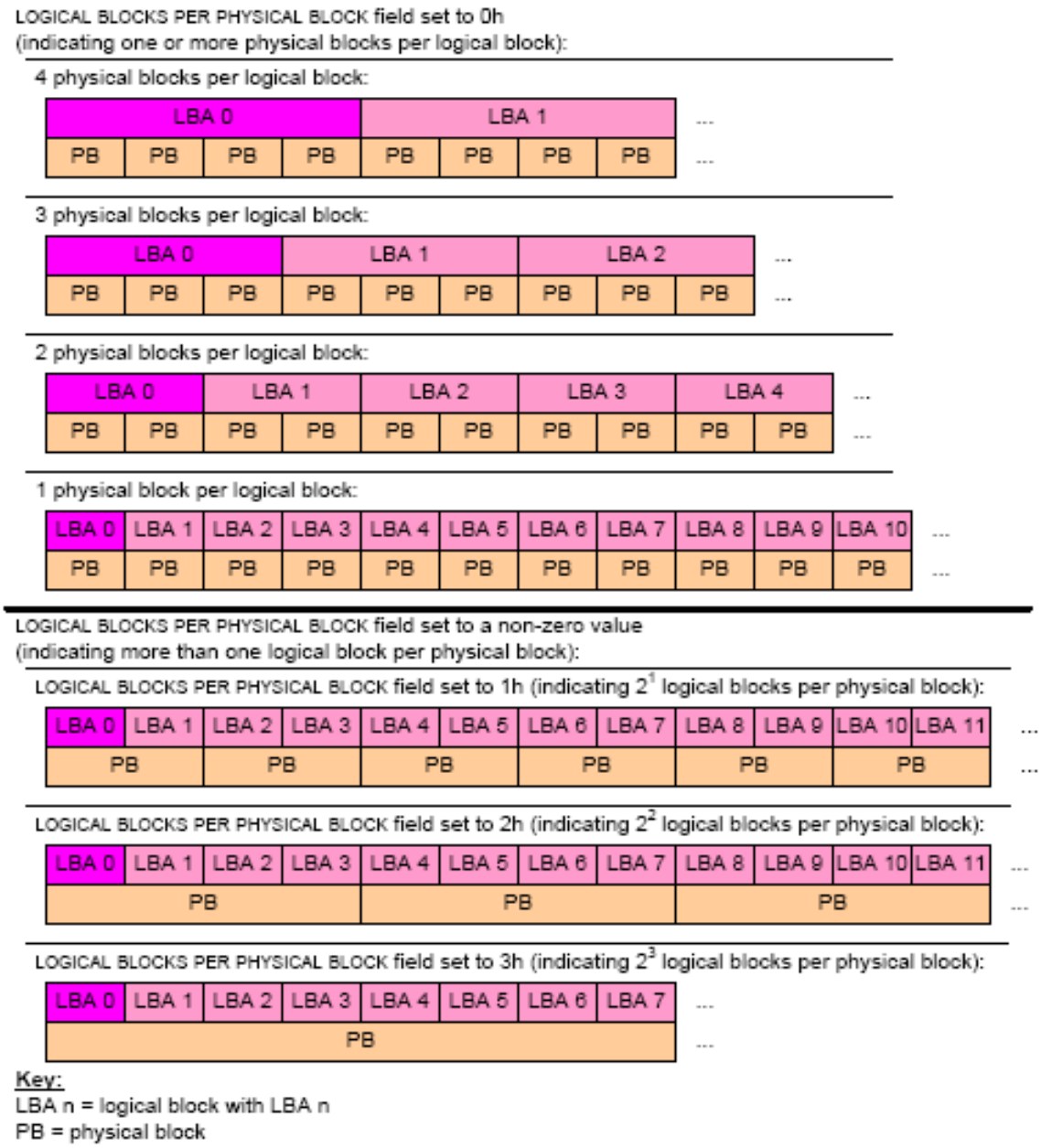

Figure 2 — Logical blocks and physical blocks examples

Figure 3 shows examples of logical blocks and physical blocks, where various LBAs are aligned to the physical block boundaries.

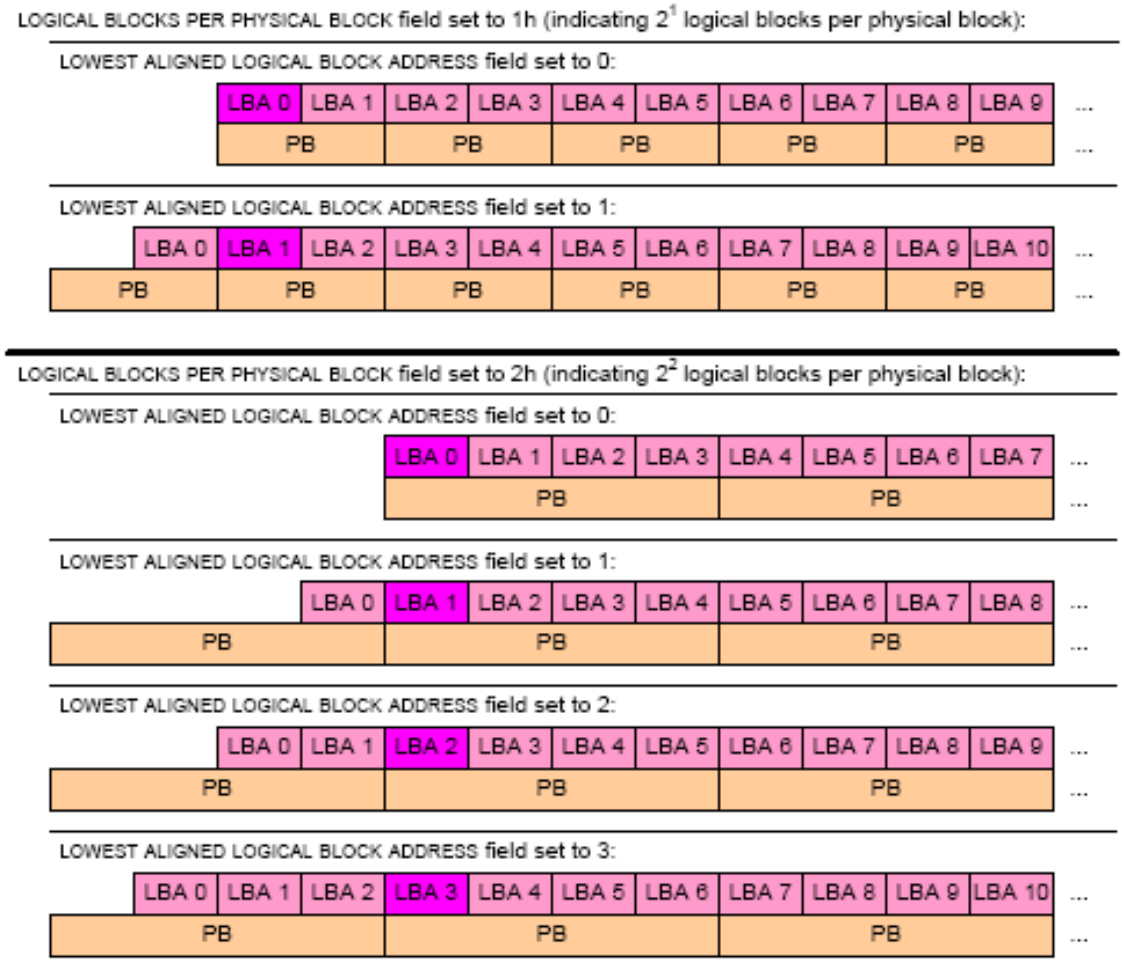

Figure 3 — Logical block to physical block alignment examples

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

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

4.6 Ready state

A direct-access block device is ready when the device server is capable of processing medium access commands (i.e., commands that perform read operations, write operations, unmap operations, or verify operations).

<...>

4.7 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 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 3 - Mode parameter block descriptors read with the MODE SENSE command before a FORMAT UNIT completes may contain 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 to be returned by a device server in response to a READ CAPACITY (10) command (see 5.12) or a READ CAPACITY (16) command (see 5.13) changes (e.g., when a FORMAT UNIT command or a MODE SELECT command completes changing the number of logical blocks, logical block length, protection information, or reference tag ownership values, or when a vendor-specific mechanism causes a change), then the device server shall establish a unit attention condition for the SCSI initiator port (see SAM-4) associated with each I_T nexus, except the I_T nexus on which the command causing the change was received, with the additional sense code set to CAPACITY DATA HAS CHANGED.

NOTE 4 - Logical units compliant with previous versions of this standard were not required to establish a unit attention condition.

4.8 Write protection

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

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

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

The device server reports the status of write protection in the device server and on the medium with the DEVICE-SPECIFIC PARAMETER field in the mode parameter header (see 6.3.1).

<...>

4.10 Write failures

If one or more commands performing write operations or unmap operations are in the task set and are being processed when power is lost (e.g., resulting in a vendor-specific command timeout by the application client) or a medium error or hardware error occurs (e.g., because a removable medium was incorrectly unmounted), then the data in the logical blocks being written or unmapped by those commands is indeterminate. When accessed by a command performing a read or verify operation (e.g., after power on or after the removable medium is mounted), the device server may return old data, new data, or vendor-specific data in those logical blocks.

Before reading or verifying logical blocks which encountered such a failure, an application client should reissue any commands performing write operations or unmap operations that were outstanding.

4.11 Caches

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

<...>

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

During write operations, the device server uses the cache to store data that is to be written to the medium at a later time. This is called write-back caching. The command may complete prior to logical blocks being written to the medium. As a result of using a write-back caching there is a period of time when the data may be lost if power to the SCSI target device is lost and a volatile cache is being used or a hardware failure occurs. There is also the possibility of an error occurring during the subsequent write operation. If an error occurred during the write operation, it may be reported as a deferred error on a later command. The application client may request that write-back caching be disabled with the Caching mode page (see 6.3.4) to prevent detected write errors from being reported as deferred errors. Even with write-back caching disabled, undetected write errors may occur. The VERIFY commands and the WRITE AND VERIFY commands may be used to detect those errors.

During unmap operations, the device server updates cache to prevent retrieval of stale data during a subsequent read operation.

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

NOTE 5 - This does not mean that stale data is allowed in the cache. If a write operation accesses the same LBA as a logical block in the cache, the logical block in the cache is updated with the new write data. If an unmap operation accesses the same LBA as a logical block in the cache, the logical block in the cache is updated with the new read data (see 4.1.3) or removed from the cache.

<…>

4.13 RESERVATIONS

Reservation restrictions are placed on commands as a result of access qualifiers associated with the type of reservation. See SPC-4 for a description of reservations. The details of commands that are allowed under what types of reservations are described in table 3.

Commands from I_T nexuses holding a reservation should complete normally. Table 3 specifies the behavior of commands from registered I_T nexuses when a registrants only or all registrants type persistent reservation is present.

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

Table 3 — SBC-2 commands that are allowed in the presence of various reservations

Command	Addressed logical unit has this type of persistent reservation held by another I_T nexus				
	From any I_T nexus		From registered I_T nexus (RR all types)	From I_T nexus not registered	
	Write Exclusive	Exclusive Access		Write Exclusive - RR	Exclusive Access - RR
FORMAT UNIT	Conflict	Conflict	Allowed	Conflict	Conflict
ORWRITE	Conflict	Conflict	Allowed	Conflict	Conflict
PRE-FETCH (10)/(16)	Allowed	Conflict	Allowed	Allowed	Conflict
PREVENT ALLOW MEDIUM REMOVAL (Prevent=0)	Allowed	Allowed	Allowed	Allowed	Allowed
PREVENT ALLOW MEDIUM REMOVAL (Prevent<>0)	Conflict	Conflict	Allowed	Conflict	Conflict
READ (6)/(10)/(12)/(16)/(32)	Allowed	Conflict	Allowed	Allowed	Conflict
READ CAPACITY (10)/(16)	Allowed	Allowed	Allowed	Allowed	Allowed
READ DEFECT DATA (10)/(12)	Allowed [a]	Conflict	Allowed	Allowed [a]	Conflict
READ LONG (10)/(16)	Conflict	Conflict	Allowed	Conflict	Conflict
REASSIGN BLOCKS	Conflict	Conflict	Allowed	Conflict	Conflict
START STOP UNIT with START bit set to one and POWER CONDITION field set to 0h	Allowed	Allowed	Allowed	Allowed	Allowed
START STOP UNIT with START bit set to zero or POWER CONDITION field set to a value other than 0h	Conflict	Conflict	Allowed	Conflict	Conflict
SYNCHRONIZE CACHE (10)/(16)	Conflict	Conflict	Allowed	Conflict	Conflict
VERIFY (10)/(12)/(16)/(32)	Allowed	Conflict	Allowed	Allowed	Conflict
WRITE (6)/(10)/(12)/(16)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE AND VERIFY (10)/(12)/(16)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE LONG (10)/(16)	Conflict	Conflict	Allowed	Conflict	Conflict
WRITE SAME (10)/(16)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
XDREAD (10)/(32)	Allowed	Conflict	Allowed	Allowed	Conflict
XDWRITE (10)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
XDWRITEREAD (10)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict
XPWRITE (10)/(32)	Conflict	Conflict	Allowed	Conflict	Conflict

Key: RR = Registrants Only or All Registrants

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

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

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

Add UNMAP command matching WRITE (6/10/12/16/32) -

UNMAP Conflict Conflict Allowed Conflict Conflict

<...>

4.17.2.1 Protection types overview

The content of protection information is dependent on the type of protection to which a logical unit has been formatted.

The type of protection supported by the logical unit shall be indicated in the SPT field in the Extended INQUIRY Data VPD page (see SPC-4). The current protection type shall be indicated in the P_TYPE field in the READ CAPACITY(16) command (see 5.13).

An application client may format the logical unit to a specific type of protection using the FMTPINFO field and the PROTECTION FIELD USAGE field in the FORMAT UNIT command (see 5.2).

The medium access commands are processed in a different manner by a device server depending on the type of protection in effect. When used in relation to types of protection, the term “medium access commands” is defined as the following commands:

a) ORWRITE;
b) READ (10);
c) READ (12);
d) READ (16);
e) READ (32);
e.1) UNMAP;
f) VERIFY (10);
g) VERIFY (12);
h) VERIFY (16);
i) VERIFY (32);
j) WRITE (10);
k) WRITE (12);
l) WRITE (16);
m) WRITE (32);
n) WRITE AND VERIFY (10);
o) WRITE AND VERIFY (12);
p) WRITE AND VERIFY (16);
q) WRITE AND VERIFY (32);
r) WRITE SAME (10);
s) WRITE SAME (16);
t) WRITE SAME (32);
u) XDWRITE (10);
v) XDWRITE (32);
w) XDWRITEREAD (10);
x) XDWRITEREAD (32);
y) XPWRITE (10);
z) XPWRITE (32);
aa) XDREAD (10); and
ab) XDREAD (32).

The device server may allow the READ (6) command (see 5.7) and the WRITE (6) command (see 5.26) regardless of the type of protection to which the logical unit has been formatted.

<...>

4.20 Association between commands and CbCS permission bits

Table 12 defines the Capability based Command Security (i.e., CbCS) permissions required for each command defined in this standard to be processed by a secure CDB processor. The permissions shown in table 12 are defined in the PERMISSIONS BIT MASK field in the capability descriptor of a CbCS extension descriptor (see SPC-4). This standard does not define any permission specific to block commands.

Table 12 — Associations between commands and CbCS permissions (part 1 of 3)

Command name	Permissions							
	DATA READ	DATA WRITE	PARM READ	PARM WRITE	SEC MGMT	RESRV	MGMT	PHY ACC
FORMAT UNIT		v		v				
ORWRITE		v						
PRE-FETCH (10)	v							
PRE-FETCH (16)	v							
Key: v = a secure CDB processor shall process the requested command only if the corresponding bit is set in the PERMISSIONS BIT MASK field in the capability descriptor of a CbCS extension descriptor (see SPC-4).								

SYNCHRONIZE CACHE (10)		v						
SYNCHRONIZE CACHE (16)		v						
VERIFY (10)	v							
VERIFY (12)	v							
VERIFY (16)	v							
VERIFY (32)	v							
WRITE (6)		v						
WRITE (10)		v						
WRITE (12)		v						
WRITE (16)		v						
WRITE (32)		v						
WRITE AND VERIFY (10)		v						

Add UNMAP command as follows (DATA WRITE permissions required):

UNMAP v

<...>

5 Commands for direct-access block devices

5.1 Commands for direct-access block devices overview

The commands for direct-access block devices are listed in table 13. Commands with CDB or parameter data fields that support protection information (see 4.17) or for which protection information may be a factor in the processing of the command are indicated by the fourth (i.e, Protection information) column.

Table 13 — Commands for direct-access block devices (part 1 of 3)

Command name	Operation code [a]	Type [b]	Protection information	Reference
ACCESS CONTROL IN	86h	O	no	SPC-4
ACCESS CONTROL OUT	87h	O	no	SPC-4
CHANGE ALIASES	A4h/0Bh	O	no	SPC-4
EXTENDED COPY	83h	O	no	SPC-4
FORMAT UNIT	04h	M	yes	5.2
INQUIRY	12h	M	yes	SPC-4
LOG SELECT	4Ch	O	no	SPC-4
LOG SENSE	4Dh	O	no	SPC-4
MAINTENANCE IN	A3h/00h to 04h A3h/06h to 09h	X [e]	no	SCC-2
MAINTENANCE OUT	A4h/00h to 05h A4h/07h to 09h	X [e]	no	SCC-2
MODE SELECT (6)	15h	O	no	SPC-4

<...>

READ CAPACITY (10)	25h	M	no	5.12
READ CAPACITY (16)	9Eh/10h	X [d]	yes	5.13
READ DEFECT DATA (10)	37h	O	no	5.14

<...>

VOLUME SET IN	BEh	X [e]	no	SCC-2
VOLUME SET OUT	BFh	X [e]	no	SCC-2
WRITE (6)	0Ah	O [c]	yes	5.26
WRITE (10)	2Ah	O	yes	5.27
WRITE (12)	AAh	O	yes	5.28
WRITE (16)	8Ah	O	yes	5.29
WRITE (32)	7Fh/000Bh	O	yes	5.30
WRITE AND VERIFY (10)	2Eh	O	yes	5.31
WRITE AND VERIFY (12)	AEh	O	yes	5.32
WRITE AND VERIFY (16)	8Eh	O	yes	5.33
WRITE AND VERIFY (32)	7Fh/000Ch	O	yes	5.34

Add UNMAP command as follows:

UNMAP	42h	X(g)	yes	5.x

(d) READ CAPACITY (16) is mandatory if protection information or thin provisioning is supported and optional otherwise.

<...>

(g) UNMAP is mandatory if thin provisioning is supported and optional otherwise.

<...>

5.4 PRE-FETCH (10) command

The PRE-FETCH (10) command (see table 32) requests that the device server:

a) for logical blocks in the mapped state, transfer the specified logical blocks from the medium to the volatile cache and/or non-volatile cache; or

b) For logical blocks in the unmapped state, update the volatile cache and/or non-volatile cache to prevent retrieval of stale data.

Logical blocks include user data and, if the medium is formatted with protection information enabled, protection information. No data shall be transferred to the data-in buffer.

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 32.

<...>

5.5 PRE-FETCH (16) command

The PRE-FETCH (16) command (see table 33) requests that the device server:

a) for logical blocks in the mapped state, transfer the specified logical blocks from the medium to the volatile cache and/or non-volatile cache; or

b) for logical blocks in the unmapped state, update the volatile cache and/or non-volatile cache to prevent retrieval of stale data.

Logical blocks include user data and, if the medium is formatted with protection information enabled, protection information. No data shall be transferred to the data-in buffer.

<...>

5.7 READ (6) command

The READ (6) command (see table 36) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes user data but does not include protection information. The most recent data value written, or to be written, if cached, in the addressed logical blocks shall be returned. For specified logical blocks that are in the unmapped state see 4.4.1.3.

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 36.

<...>

5.8 READ (10) command

The READ (10) command (see table 38) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes user data and may include protection information, based on the RDPROTECT field and the medium format. The most recent data value written or to be written, if cached, in the addressed logical block shall be returned. For specified logical blocks that are in the unmapped state see 4.4.1.3.

<...>

5.9 READ (12) command

The READ (12) command (see table 41) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes user data and may include protection information, based on the

RDPROTECT field and the medium format. The most recent data value written or to be written, if cached, in the addressed logical block shall be returned. For specified logical blocks that are in the unmapped state see 4.4.1.3.

<...>

5.10 READ (16) command

The READ (16) command (see table 42) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes user data and may include protection information, based on the RDPROTECT field and the medium format. The most recent data value written or to be written, if cached, in the addressed logical block shall be returned. For specified logical blocks that are in the unmapped state see 4.4.1.3.

<...>

5.11 READ (32) command

The READ (32) command (see table 43) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes user data and may include protection information, based on the RDPROTECT field and the medium format. The most recent data value written or to be written, if cached, in the addressed logical block shall be returned. For specified logical blocks that are in the unmapped state see 4.4.1.3.

The READ (32) command shall only be processed if type 2 protection is enabled (see 4.17.2.4).

<...>

5.13 READ CAPACITY (16) command

5.13.1 READ CAPACITY (16) command overview

The READ CAPACITY (16) command (see table 46) requests that the device server transfer parameter data describing the capacity and medium format of the direct-access block device to the data-in buffer. This command is mandatory if the logical unit supports protection information (see 4.17) and is optional otherwise. This command is implemented as a service action of the SERVICE ACTION IN operation code (see A.2). This command may be processed as if it has a HEAD OF QUEUE task attribute (see 4.12).

Table 46 – READ CAPACITY (16) command

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

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

See the READ CAPACITY (10) command (see 5.12) for definitions of the LOGICAL BLOCK ADDRESS field and the PMI bit.

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

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

5.13.2 READ CAPACITY (16) parameter data

The READ CAPACITY (16) parameter data is defined in table 47. Any time the READ CAPACITY (16) parameter data changes, the device server should establish a unit attention condition as described in 4.7.

Table 47 – READ CAPACITY (16) parameter data

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

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

The protection type (P_TYPE) field and the protection enable (PROT_EN) bit (see table 48) indicate the logical unit's current type of protection.

Table 48 — P_TYPE field and PROT_EN bit

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

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

Table 49 — LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field

Code	Description
0	One or more physical blocks per logical block [a]
n > 0	2^n logical blocks per physical block

[a] The number of physical blocks per logical block is not reported.

A thin provisioning enabled (TPE) bit set to one indicates that the logical unit implements thin provisioning (see 4.4.1.5).

A TPE bit set to zero indicates that the logical unit implements full provisioning (see 4.4.1.4).

A thin provisioning read zeros (TPRZ) bit set to one indicates that a read operation or verify operation of an LBA that is in the unmapped state ~~will~~ shall ~~transfer~~ retrieve data with all bits set to zero ~~to the data-in buffer~~.

A TPRZ bit set to zero indicates that a read operation or verify operation of an LBA that is in the unmapped state may ~~transfer~~ retrieve bits of other than all zeros ~~to the data-in buffer~~.

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

NOTE 14 - The highest LBA that the lowest aligned logical block address field supports is 3FFFh (i.e., 16383).

<...>

5.18 REASSIGN BLOCKS command

5.18.1 REASSIGN BLOCKS command overview

The REASSIGN BLOCKS command (see table 56) 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 4.9).

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, then the device server shall write the recovered user data and any protection information to the reassigned logical block. If the LBA is in the unmapped state, the device server shall transition the LBA to the mapped state and write the data retrieved during a read operation (see 4.4.1.3) to the reassigned logical block. If the device server is unable to recover user data and protection information, if any, then the device server shall write vendor-specific data as the user data and shall write a default value of FFFF_FFFF_FFFF_FFFFh as the protection information, if enabled. The data in all other logical blocks on the medium shall be preserved.

<...>

5. x UNMAP command

5. x. 1 UNMAP command overview

The UNMAP command specifies that the device server should transition one or more logical blocks to the unmapped state. The UNMAP command shall be implemented by device servers supporting thin provisioning (see 4.4.1.5).

Table x.1 – UNMAP Command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (42h)							
1	Reserved							
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved			GROUP NUMBER				
7	(MSB) PARAMETER LIST LENGTH							
8	(LSB)							
9	Control							

The OPERATION CODE field is defined in SPC-4 shall be set to the value defined in table x.1.

See the PRE-FETCH (10) command (see 5.4) and 4.18 for the definition of the GROUP NUMBER field.

The PARAMETER LIST LENGTH field specifies the length in bytes of the UNMAP PARAMETER LIST that shall be transferred from the application client to the device server. A PARAMETER LIST LENGTH of zero specifies that no data shall be transferred.

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

For each specified LBA:

a) a mapped LBA should be unmapped, or may remain mapped; and
b) an unmapped LBA shall remain unmapped.

5. x. 2 UNMAP parameter list

The UNMAP parameter list (see table x.3) contains a~~n eight byte~~ parameter list header followed by a UNMAP DESCRIPTOR LIST containing one or more UNMAP LBA DESCRIPTOR fields.

Table x.3 – UNMAP parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	UNMAP DESCRIPTOR LIST LENGH (n-3)							
1								
2	Reserved							
7								
UNMAP DESCRIPTOR LIST								
8	UNMAP LBA DESCRIPTOR							
23								
…	…							
n-15	UNMAP LBA DESCRIPTOR							
n								

The UNMAP DESCRIPTOR LIST LENGTH describes the length of the UNMAP DESCRIPTOR LIST.

The UNMAP DESCRIPTOR LIST contains a list of LBA extents to be operated on. The UNMAP LBA DESCRIPTOR is described in table x.4. The LBAs in the UNMAP DESCRIPTOR LIST may contain overlapping extents, and may be in any order. If the number of LBAs exceeds the allowed number, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Table x.4 – UNMAP LBA DESCRIPTOR

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB) LBA							
7								(LSB)
8	(MSB) LBA COUNT							
11								(LSB)
12	RESERVED							
15								

If the LBA plus the ~~count~~ LBA COUNT exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE.

An LBA COUNT field set to zero shall not be considered an error.

<...>

5.22 VERIFY (10) command

The VERIFY (10) command (see table 66) requests that the device server verify the specified logical block(s) on the medium. Each logical block includes user data and may include protection information, based on the VRPROTECT field and the medium format.

Table 66 — VERIFY (10) command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Fh)							
1	VRPROTECT			DPO	Reserved		BYTCHK	Obsolete
2	(MSB)							
5	LOGICAL BLOCK ADDRESS							(LSB)
6	Restricted for MMC-6	Reserved		GROUP NUMBER				
7	(MSB)							
8	VERIFICATION LENGTH							(LSB)
9	CONTROL							

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 5.20 and 5.21) 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 BLOCKS field set to the value of the VERIFY command's VERIFICATION LENGTH field).

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 66.

See the READ (10) command (see 5.8) for the definition of the DPO bit. See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.4) and 4.18 for the definition of the GROUP NUMBER field.

If the Verify Error Recovery mode page (see 6.3.6) is implemented, then the current settings in that page specify the verification criteria. If the Verify Error Recovery mode page is not implemented, then the verification criteria is vendor-specific.

For LBAs in the mapped state:

a) if the byte check (BYTCHK) bit is set to zero, then the device server shall:

   A) perform a medium verification with no data comparison and not transfer any data from the data-out buffer; and
   B) check protection information read from the medium based on the VRPROTECT field as described in table 67.

and

b) if the BYTCHK bit is set to one, then the device server shall:

   A) perform a byte-by-byte comparison of user data read from the medium and user data transferred from the data-out buffer;
   B) check protection information read from the medium based on the VRPROTECT field as described in table 68;
   C) check protection information transferred from the data-out buffer based on the VRPROTECT field as described in table 69; and
   D) perform a byte-by-byte comparison of protection information read from the medium and transferred from the data-out buffer based on the VRPROTECT field as described in table 70.

For LBAs in the unmapped state:

a) If the BYTCHK bit is set to zero, then the device server shall complete the command with GOOD status; and

b) If the BYTCHK bit is set to one, then the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The order of the user data and protection information checks and comparisons is vendor-specific.

If a byte-by-byte comparison is unsuccessful for any reason, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to MISCOMPARE and the additional sense code set to the appropriate value for the condition.

The VERIFICATION LENGTH field specifies the number of contiguous logical blocks that shall be verified, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. If the BYTCHK bit is set to one, then the VERIFICATION LENGTH field also specifies the number of logical blocks that the device server shall transfer from the data-out buffer. A VERIFICATION LENGTH field set to zero specifies that no logical blocks shall be verified. This condition shall not be considered as an error. Any other value specifies the number of logical blocks that shall be verified. If the LBA plus the verification length exceeds the capacity of the medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to LOGICAL BLOCK ADDRESS OUT OF RANGE. The VERIFICATION LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.4.2).

<...>

6.3 Mode parameters

<...>

6.3.5 Read-Write Error Recovery mode page

The Read-Write Error Recovery mode page (see table 127) specifies the error recovery parameters the device server shall use during any command that performs a read or write operation to the medium (e.g., READ command, WRITE command, or a WRITE AND VERIFY command).

Table 127 — Read-Write Error Recovery mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF(0b)	PAGE CODE (01h)					
1	PAGE LENGTH (0Ah)							
2	AWRE	ARRE	TB	RC	Error Recovery Bits			
					EER	PER	DTE	DCR
3	READ RETRY COUNT							
4	OBSOLETE							
5	OBSOLETE							
6	OBSOLETE							
7	TPNER	RESERVED					Restricted for MMC-6	
9	WRITE RETRY COUNT							
8	Reserved							
10	(MSB) RECOVERY TIME LIMIT							
11	(LSB)							

The parameters savable (PS) bit is only used with the MODE SENSE command. This bit is reserved with the MODE SELECT command. A PS bit set to one indicates that the device server is capable of saving the mode page in a non-volatile vendor-specific location.

The SubPage Format (SPF) bit, PAGE CODE field, and PAGE LENGTH field are defined in SPC-4 and shall be set to the values defined in table 127.

An automatic write reallocation enabled (AWRE) bit set to zero specifies that the device server shall not perform automatic reallocation of defective logical blocks during write operations.

An AWRE bit set to one specifies that the device server shall enable automatic reallocation of defective logical blocks during write operations. The automatic reallocation shall be performed only if the device server has the valid data (e.g., original data in a buffer or recovered from the medium). The valid data shall be placed in the reallocated logical block. The device server shall report any failures that occur during the reallocation operation. Error reporting as specified by the error recovery bits (i.e., the EER bit, the PER bit, the DTE bit, and the DCR bit) shall be performed only after completion of the reallocation. See the REASSIGN BLOCKS command (see 5.18) for error procedures.

An automatic read reallocation enabled (ARRE) bit set to zero specifies that the device server shall not perform automatic reallocation of defective logical blocks during read operations.

An ARRE bit set to one specifies that the device server shall enable automatic reallocation of defective logical blocks during read operations. All error recovery actions required by the error recovery bits (i.e., the EER bit, the PER bit, the DTE bit, and the DCR bit) shall be processed. The automatic reallocation shall then be performed only if the device server successfully recovers the data. The recovered data shall be placed in the reallocated logical block. The device server shall report any failures that occur during the reallocation operation. Error reporting as specified by the

error recovery bits (i.e., the EER bit, the PER bit, the DTE bit, and the DCR bit) shall be performed only after completion of the reallocation operation. See the REASSIGN BLOCKS command (see 5.18) for error procedures.

A transfer block (TB) bit set to zero specifies that if an unrecovered read error occurs during a read operation, then the device server shall not transfer any data for the logical block to the data-in buffer. A TB bit set to one specifies that if an unrecovered read error occurs during a read operation, then the device server shall transfer pseudo read data before returning CHECK CONDITION status. The data returned in this case is vendor-specific. The TB bit does not affect the action taken for recovered read errors.

A read continuous (RC) bit set to zero specifies that error recovery operations that cause delays during the data transfer are acceptable. Data shall not be fabricated.

An RC bit set to one specifies the device server shall transfer the entire requested length of data without adding delays during the data transfer to perform error recovery procedures. The device server may transfer pseudo read data in order to maintain a continuous flow of data. The device server shall assign priority to the RC bit over conflicting bits within this byte.

NOTE 24 - The RC bit may be set to one in image processing, audio, or video applications.

An enable early recovery (EER) bit set to one specifies that the device server shall use the most expedient form of error recovery first. An EER bit set to zero specifies that the device server shall use an error recovery procedure that minimizes the risk of error mis-detection or mis-correction. This bit only applies to data error recovery and it does not affect positioning retries.

NOTE 25 - An EER bit set to one may imply an increase in the probability of error mis-detection or mis-correction. An EER bit set to zero allows the specified retry limit to be exhausted prior to using additional information (e.g., ECC bytes) to correct the error.

A post error (PER) bit set to one specifies that if a recovered read error occurs during a command performing a read or write operation, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR. A PER bit set to zero specifies that if a recovered read error occurs during a command performing a read or write operation, then the device server shall perform error recovery procedures within the limits established by the error recovery parameters and only terminate the command with CHECK CONDITION status if the error becomes uncorrectable based on the established limits. If the DTE bit is set to one, then the PER bit shall be set to one.

A data terminate on error (DTE) bit set to one specifies that the device server shall terminate the data-in or data-out buffer transfer of a command performing a read or write operation upon detection of a recovered error. A DTE bit set to zero specifies that the device server shall not terminate the data-in or data-out buffer transfer of a command performing a read or write operation upon detection of a recovered error.

A disable correction (DCR) bit set to one specifies that additional information (e.g., ECC bytes) (see 4.4) shall not be used for data error recovery. A DCR bit set to zero allows the use of additional information (e.g., ECC bytes) for data error recovery. If the EER bit is set to one, the DCR bit shall be set to zero.

The combinations of the error recovery bits (i.e., the EER bit, the PER bit, the DTE bit, and the DCR bit) are explained in table 128.

<...>

The READ RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during read operations.

A thin provisioning no error report (TPNER) bit set to one specifies that if a thin provisioning soft threshold is crossed, that a UNIT ATTENTION, with the additional sense code set to PROVISIONING SOFT THRESHOLD REACHED shall not be queued. A TPNER bit set to zero specifies that if the thin provisioning soft threshold is crossed, a UNIT ATTENTION with the additional sense code set to PROVISIONING SOFT THRESHOLD REACHED shall be queued.

The WRITE RETRY COUNT field specifies the number of times that the device server shall attempt its recovery algorithm during write operations.

The RECOVERY TIME LIMIT field specifies in milliseconds the maximum time duration that the device server shall use for data error recovery procedures. The device server may round this value as described in SPC-4. The limit in this field specifies the maximum error recovery time allowed for any individual logical block. A RECOVERY TIME LIMIT field set to zero specifies that the device server shall use its default value.

When both a retry count and a recovery time limit are specified, the field that specifies the recovery action of least duration shall have priority.

NOTE 26 - To disable all types of correction and retries the application client should set the EER bit to zero, the PER bit to one, the DTE bit to one, the DCR bit to one, the READ RETRY COUNT field to 00h, the WRITE RETRY COUNT field to 00h, and the RECOVERY TIME LIMIT field to 0000h.

6.4 Vital product data (VPD) parameters

6.4.1 VPD parameters overview

<...>

6.4.2 Block Limits VPD page

The Block Limits VPD page (see table 132) provides the application client with the means to obtain certain operating parameters of the logical unit.

Table 132 — Block Limits VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (B0h)							
2	Reserved							
3	PAGE LENGTH (14h)							
4	Reserved							
5								
6	(MSB) OPTIMAL TRANSFER LENGTH GRANULARITY							
7	(LSB)							
8	(MSB) MAXIMUM TRANSFER LENGTH							
11	(LSB)							

12	(MSB)
15	OPTIMAL TRANSFER LENGTH (LSB)
16	(MSB)
19	MAXIMUM PREFETCH XDREAD XDWRITE TRANSFER LENGTH (LSB)
20	(MSB)
23	MAXIMUM UNMAP LBA COUNT (LSB)

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in SPC-4.

The PAGE CODE field and PAGE LENGTH field are defined in SPC-4 and shall be set to the values defined in table 132.

The OPTIMAL TRANSFER LENGTH GRANULARITY field indicates the optimal transfer length granularity in blocks for a single ORWRITE command, PRE-FETCH command, READ command, VERIFY command, WRITE command, WRITE AND VERIFY command, XDREAD command, XDWRITE command, XDWRITEREAD command, or XPWRITE command. Transfers with transfer lengths not equal to a multiple of this value may incur significant delays in processing.

The MAXIMUM TRANSFER LENGTH field indicates the maximum transfer length in blocks that the device server accepts for a single ORWRITE command, READ command, VERIFY command, WRITE command, WRITE AND VERIFY command, XDWRITEREAD command, or XPWRITE command. Requests for transfer lengths exceeding this limit result in CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB. A MAXIMUM TRANSFER LENGTH field set to zero indicates that there is no reported limit on the transfer length.

The OPTIMAL TRANSFER LENGTH field indicates the optimal transfer length in blocks for a single ORWRITE command, PRE-FETCH command, READ command, VERIFY command, WRITE command, WRITE AND VERIFY command, XDREAD command, XDWRITE command, XDWRITEREAD command, or XPWRITE command. Transfers with transfer lengths exceeding this value may incur significant delays in processing.

The MAXIMUM PREFETCH XDREAD XDWRITE TRANSFER LENGTH field indicates:

a) the maximum transfer length in blocks that the device server accepts for a single PRE-FETCH command;
b) if the XOR Control mode page (see 6.3.7) is implemented, then the maximum value supported by the MAXIMUM XOR WRITE SIZE field in the XOR Control mode page; and
c) if the XOR Control mode page is not implemented, then the maximum transfer length in blocks that the device server accepts for a single XDWRITE command or XDREAD command.

The device server should set the MAXIMUM PREFETCH XDREAD XDWRITE TRANSFER LENGTH field to less than or equal to the MAXIMUM TRANSFER LENGTH field.

The MAXIMUM UNMAP LBA COUNT field indicates the maximum number of LBAs that may be represented in the parameter data transferred to the device server in the UNMAP PARAMETER LIST (see 5.x.2). If the logical unit implements thin provisioning, then this value shall be greater than or equal to one.

SPC4 Changes

7.2.13 Statistics and Performance log pages

7.2.13.1 Statistics and Performance log pages overview

The Statistics and Performance log pages consist of a General Statistics and Performance log page and up to 31 Group Statistics and Performance log pages. Each Group Statistics and Performance log pages only collects statistics and performance information for the group number specified in a read CDB or a write CDB.

The General Statistics and Performance log page (see 7.2.13.2) provides the following statistics and performance results associated to the addressed logical unit:

a) Number of read commands;
b) Number of write commands;
c) Number of read logical blocks transmitted by a target port;
d) Number of write logical blocks received by a target port;
e) Read command processing time;
f) Write command processing time;
g) Sum of the command weights of the read commands plus write commands;
h) Sum of the weighted command time of the read commands plus write commands;
i) Idle time; and
j) Time interval.

The Group Statistics and Performance log pages (see 7.2.13.3) provide the following statistics and performance results associated to the addressed logical unit and the GROUP NUMBER field:

a) Number of read commands;
b) Number of write commands;
c) Number of read logical blocks transmitted by a target port;
d) Number of write logical blocks received by a target port;
e) Read command processing time; and
f) Write command processing time.

In the Statistics and Performance log pages, read and write commands are those shown in table 309.

Table 309 – Statistics and Performance log pages read and write commands

Read commands (a)	**Write commands(a)**
READ(6) READ(10) READ(12) READ(16) READ(32)	UNMAP WRITE(6) WRITE(10) WRITE(12) WRITE(16) WRITE(32) WRITE AND VERIFY (10) WRITE AND VERIFY (12) WRITE AND VERIFY (16) WRITE AND VERIFY (32)
(a) See SBC-3.	