To: INCITS Technical Committee T10
From: Fred Knight, Network Appliance
Email: knight@netapp.com
Date: Nov 4, 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 UMAP 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 UMAP command to the blocks VPD page, and account for XCOPY (SPC).

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	ASC/Q
Temporary lack of physical blocks – write not done, retry required	NOT READY	SPACE ALLOCATION IN PROCESS
Persistent lack of physical blocks – write not done, retry will not help	DATA PROTECT	SPACE ALLOCATION FAILED
Soft Threshold crossed – write not done, retry required	UNIT ATTENTION	PROVISIONING SOFT THRESHOLD REACHED (*)
(*) – a unit attention condition is also established for the initiator port associated with every I_T nexus with the ASC/Q set to PROVISIONING SOFT THRESHOLD REACHED.		

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:

Glossary entry for mapped and unmapped added – also search for other references elsewhere – no other references.

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 a LBA in which there exists a known relationship to a physical block.

3.1.33 media: Plural of medium.

3.1.34 medium: The material on which data is stored (e.g., a magnetic disk).

<…>

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 a LBA in which this standard does not define a relationship to a physical block.

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 (e.g., WRITE(10), WRITE AND VERIFY(16), UMAP) causes one or more logical blocks to be written to the medium. A read operation (e.g., READ(6)) causes one or more logical blocks to be read from the medium. A verify operation (e.g., VERIFY(16)) 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.

<…>

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 logical block may be mapped or unmapped. For logical blocks that are mapped, there is a known relationship to a physical block. For logical blocks that are unmapped, this standard does not specify the relationship between a logical block and a physical block. Figure a shows the relationships of physical blocks to logical blocks in these two states.

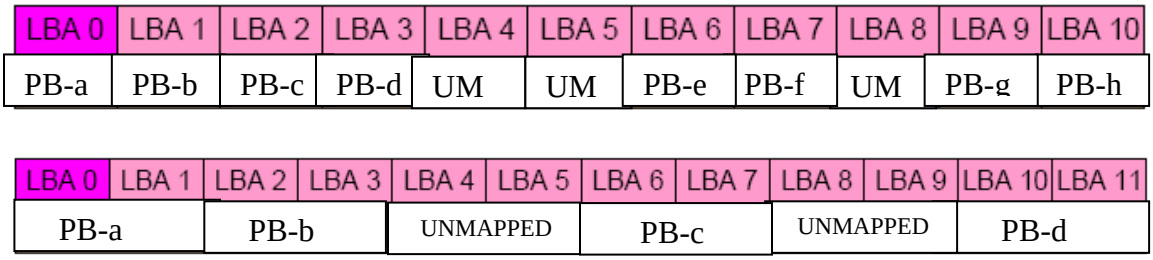


Figure a – Mapped and Unmapped Logical Blocks

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

.

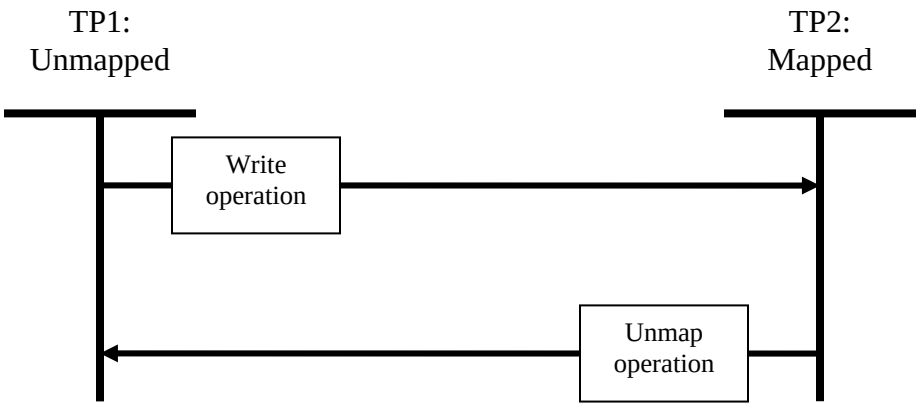


Figure b – Logical Block State Transitions

4.4.1.2 TP2: Mapped state

When in the mapped state, an association exists between the LBA and a PB. Data and protection information read from a LBA in the mapped state shall be the data and protection information that was most recently written to that LBA. If data has not been written to that LBA, data and protection information read from the LBA shall be the data and protection information that was established during initialization (see 4.7).

4.4.1.2.1 Transition TP1:Unmapped state to TP2:Mapped state

This transition shall occur when a successful write operation occurs. The transition of one LBA to the mapped state may also cause other LBAs to also transition to the mapped state (see figure a). A logical unit may cause a LBA to transition from unmapped to mapped for vendor specific reasons.

4.4.1.3 TP1: Unmapped state

When in the unmapped state, no association exists between the LBA and a PB (see figure a). Data and protection information provided during a read operation or used during a verify operation from a LBA in the unmapped state:

a) shall not be retrieved from data that was previously written to any other LBA, and
b) shall not change until a subsequent write command to that logical block successfully completes (i.e., all read operations to a logical block shall return the same data until a subsequent write operation to that logical block successfully completes).

4.4.1.3.1 Transition TP2:Mapped state to TP1:Unmapped state

This transition occurs when a successful unmap operation occurs (see 4.4.1.5.2).

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 association 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.

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 logical blocks available to store user data. 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 logical block on a thin provisioned logical unit is unmapped (see figure b).

A device server that supports thin provisioning may also support a soft threshold value. The method of setting the soft threshold value 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 UNMAP command (see 5.x.1),
b) set the TPE bit to one in the READ CAPACITY(16) parameter data (see 6.4.3), and
c) set the MAXIMUM UNMAP PARAMETER LIST LENGTH field in the Block Device Characteristics VPD page (see 6.4.3).

4.4.1.5.2 UNMAP Operation

Match “extents” with existing SPC/SBC usage (extent is already in the definitions). Replace punched w/unmapped.

An UNMAP operation transitions one or more LBA extents to the unmapped state. The protection information (if present) contained in the blocks that are unmapped shall be set to FFFF_FFFF_FFFF_FFFFh. More than one physical block may be affected by an unmapped operation. The data in all other logical blocks on the medium shall be preserved.

It is not an error for a UNMAP command to operate on an LBA that has already been unmapped.

The device server may cause different transitions for each logical block in the extent list.

4.4.1.5.3 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 the soft threshold value, 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.

4.4.1.5.4 Thin Provisioning and Protection Information

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.

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. The LOWEST ALIGNED LOGICAL BLOCK ADDRESS field and LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT of the READ CAPACITY (16) parameter data (see 5.13.2) may be used by an application client to determine the logical block alignment.

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

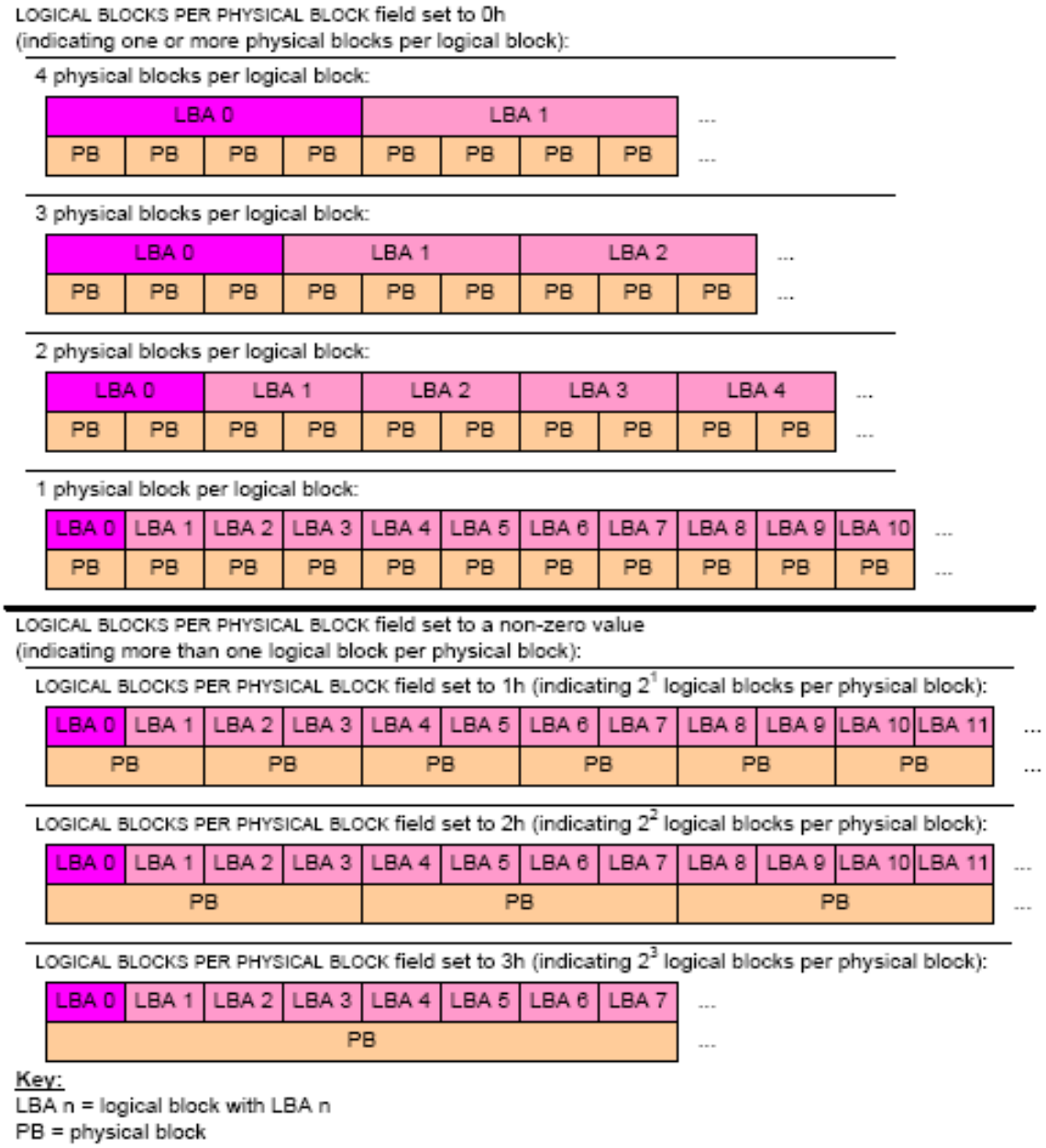
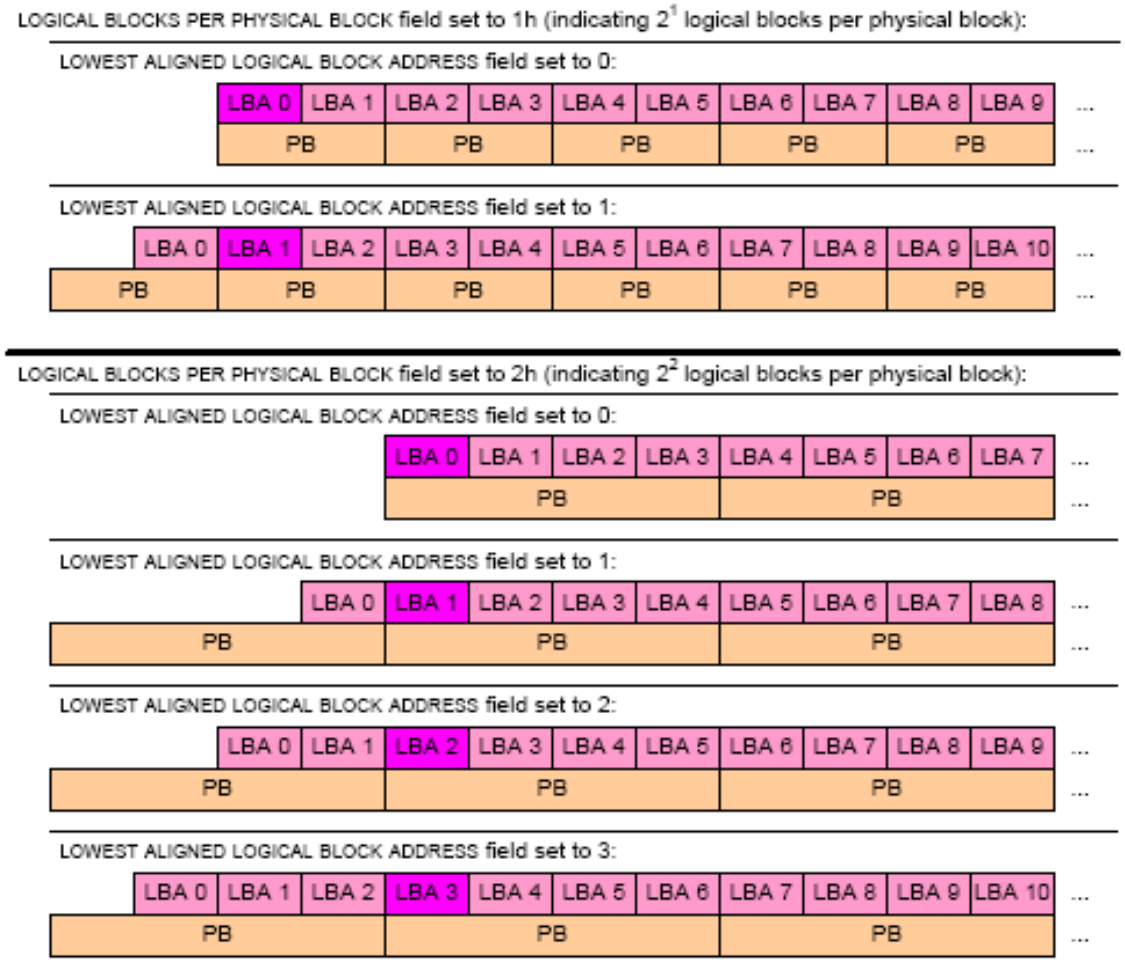


Figure 2 — Logical blocks and physical blocks examples

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



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

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.

<...>

4.7 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 commands that request alteration of the medium.

Unmap requests an 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).

<…>

4.13 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):

UMAP 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

<...>

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	O(g)	yes	5.x

(g) UNMAP is mandatory if the logical unit supports thin provisioning and optional otherwise.

<...>

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	TPE	Reserved			P_TYPE			PROT_EN
13	Reserved				LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT			
14	Reserved		(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 TPE (thin provisioning enabled) 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).

Should the TPRZ bit be here as well?

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. x UNMAP command

5. x. 1 UNMAP command overview

The UNMAP command shall be implemented by device servers supporting thin provisioning (see 4.4.1.5). The UNMAP command requests alteration of the medium. The UNMAP command (see table x.1) provides information to the device server that may be used by the device server to transition specified ranges of blocks to the unmapped state.

Table x.1 – UNMAP Command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (42h)							
1	Reserved							IMMED?
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved			GROUP NUMBER(needed?)				
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. Do we need an IMMED bit or Group Number field (modify SPC to add UNMAP to the group WRITE statistic)?

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.

5. x. 2 UNMAP parameter list

The UNMAP parameter list (see table x.3) contains an 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 LBA descriptors 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 TOO MANY SEGMENT DESCRIPTORS.

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 exceeds the capacity of the medium, it 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.

Verify operations that operate on LBAs that are in the unmapped state shall be performed using user data (see 4.4.1.3) and protection information (see 4.4.1.5.4) that would be returned if a read operation were performed on that LBA.

Table 66 — VERIFY (10) command

Bit / Byte	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							

<...>

6.4 Vital product data (VPD) parameters

6.4.1 VPD parameters overview

<...>

6.4.3 Block Device Characteristics VPD page

The Block Device Characteristics VPD page contains parameters indicating characteristics of the logical unit. Table 133 defines the Block Device Characteristics VPD page.

TABLE 133 – Block Device Characteristics VPD Page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (B1h)							
2	Reserved							
3	PAGE LENGTH (3Ch)							
4	MEDIUM ROTATION RATE							
5								
6	Reserved							
7	Reserved				NOMINAL FORM FACTOR			
8	Reserved							TPRZ
9	(MSB) MAXIMUM UNMAP PARAMETER LIST LENGTH							
10								(LSB)
11	Reserved							
63								

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in SPC-4. The PAGE CODE field and PAGE LENGTH field are defined in SPC-4 and shall be set to the values defined in table 133.

The MEDIUM ROTATION RATE field is defined in table 134.

Table 134 — MEDIUM ROTATION RATE field

Code	Description
0000h	Medium rotation rate is not reported
0001h	Non-rotating medium (e.g., solid state)
0002h to 0400h	Reserved
0401h to FFFEh	Nominal medium rotation rate in rotations per minute (i.e., rpm) (e.g., 7 200 rpm = 1C20h, 10 000 rpm = 2710h, and 15 000 rpm = 3A98h)
FFFFh	Reserved

The NOMINAL FORM FACTOR field indicates the nominal form factor of the device containing the logical unit and is defined in table 135.

Table 135 — NOMINAL FORM FACTOR field

Code	Description
0h	Nominal form factor is not reported
1h	5.25 inch
2h	3.5 inch
3h	2.5 inch
4h	1.8 inch
5h	Less than 1.8 inch
All others	Reserved

Should TPRZ be here or in READ CAPACITY?

A TPRZ (thin provisioning read zeros) bit set to one indicates that the data returned for a read operation of a LBA that is in the unmapped state will return zeros.

A TPRZ bit set to zero indicates that the data returned for a read operation of a LBA that is in the unmapped state is not defined by this standard.

The MAXIMUM UNMAP PARAMETER LIST LENGTH field indicates the maximum number of bytes that may be sent 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 24.

<...>

SPC4r16 Changes

6.3 EXTENDED COPY command

6.3.1 EXTENDED COPY command introduction

The EXTENDED COPY command (see table 99) provides a means to copy data from one set of logical units to another set of logical units or to the same set of logical units. The entity within a SCSI device that receives and performs the EXTENDED COPY command is called the copy manager. The copy manager is responsible for copying data from the source devices to the destination devices. The copy source and destination devices are logical units that may reside in different SCSI devices or the same SCSI device. It is possible that the copy source device, copy destination device, and the copy manager are the same logical unit.

<…>

6.3.3 Errors detected during processing of segment descriptors

Errors may occur after the copy manager has begun processing segment descriptors. These errors include invalid parameters in segment descriptors, invalid segment descriptors, unavailable targets referenced by target descriptors, inability of the copy manager to continue operating, and errors reported by source or destination copy target devices. If the copy manager receives CHECK CONDITION status from one of the copy target devices, it shall recover the sense data associated with the exception condition and clear any ACA condition associated with the CHECK CONDITION status.

If it is not possible to complete processing of a segment because the copy manager is unable to establish communications with a copy target device, because the copy target device does not respond to INQUIRY, or because the data returned in response to INQUIRY indicates an unsupported logical unit, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY TARGET DEVICE NOT REACHABLE.

If it is not possible to complete processing of a segment because the data returned in response to an INQUIRY command indicates a device type that does not match the type in the target descriptor, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to INCORRECT COPY TARGET DEVICE TYPE.

If the copy manager has issued a command other than INQUIRY to a copy target device while processing an EXTENDED COPY command and the copy target device either fails to respond with status or responds with status other than RECOVERED ERROR, or one listed in table Q, then the condition shall be considered a copy target device command failure. In response to a copy target device command failure the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to THIRD PARTY DEVICE FAILURE.

If a copy target device completes a command from the copy manager with a status listed in table Q, then the copy manager shall either retry the command or terminate the EXTENDED COPY command as a copy target device command failure. If the copy target device completes a command from the copy manager with a status of CHECK CONDITION listed in table Q, then the copy manager should retry the command (see SBC).

Table Q – Special Copy Target Device Responses

Status	Sense Key	Additional Sense Code
BUSY	n/a	n/a
TASK SET FULL	n/a	n/a
ACA ACTIVE	n/a	n/a
RESERVATION CONFLICT	n/a	n/a
CHECK CONDITION	NOT READY	SPACE ALLOCATION IN PROCESS
CHECK CONDITION	UNIT ATTENTION	THIN PROVISIONING SOFT THRESHOLD REACHED

NOTES

23 The copy manager is assumed to employ a vendor specific retry policy that minimizes time consuming and/or fruitless repetition of retries.

24 RESERVATION CONFLICT status is listed only to give the copy manager leeway in multi-port cases. The copy manager may have multiple initiator ports that are capable of reaching a copy target device, and a persistent reservation may restrict access to a single I_T nexus. The copy manager may need to try access from multiple initiator ports to find the correct I_T nexus.

If a copy target device responds to an input or output operation with a GOOD status but less data than expected is transferred, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY TARGET DEVICE DATA UNDERRUN. If an overrun is detected, then the EXTENDED COPY command shall be terminated with CHECK CONDITION status, with the sense key set to COPY ABORTED, and the additional sense code set to COPY TARGET DEVICE DATA OVERRUN.

<…>