NetApp™
Go further, faster

To: INCITS Technical Committee T10
From: Fred Knight, Network Appliance
Email: knight@netapp.com
Date: Sept 5, 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 terminology.
Revision 2 (Sept 5, 2008) Revision 2
Move text from the command clause to the model clause and clarify some language.

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 accepted 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 RECOVERED ERROR status if the I/O could actually succeed (or a UNIT ATTENTION if the I/O needed 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	UNIT ATTENTION	SPACE ALLOCATION IN PROCESS
Persistent lack of physical blocks – write not done, retry will not help	HARDWARE ERROR	NO ADDITIONAL SPACE CAN BE ALLOCATED
Soft Threshold crossed – write not done, retry required	UNIT ATTENTION	PROVISIONING SOFT THRESHOLD REACHED (*)

Soft Threshold crossed – write done	RECOVERED ERROR	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 the 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 Trim command.

This proposal defines commands and error codes for the operation of thinly provisioned devices. The Trim 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:

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 <or - Capacity Management>

4.4.1.1 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 units reported capacity (see 5.12 - READ CAPACITY (10) and 5.13 - READ CAPACITY(16)).

4.4.1.2 Thin Provisioning

4.4.1.2.1 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 non-volatile medium. 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.

4.4.1.2.2 Trimming Unused Logical Blocks

A TRIM operation indicates to the device server that the application client no longer needs the user data contents of a logical block. The user data and protection information contained in the blocks specified in the TRIM PARAMETER LIST of the TRIM command (5.x) may be discarded. More than one physical block may be trimmed by each TRIM LBA DESCRIPTOR. The data in all other logical blocks on the medium shall be preserved. The user data and protection information read from an LBA that has been trimmed and has not been rewritten, shall be either:

a) any constant value (e.g., all zeros, all ones) with protection information of FFFF_FFFF_FFFF_FFFFh, or
b) the previous user data contents and protection information that existed prior to processing the TRIM command.

It is not an error for a TRIM command to operate on an LBA that has already been trimmed. It is not an error for the device server to not discard user data and protection information contained in some or all of the specified LBAs <**Note:** case b in the above list>.

4.4.1.2.3 Resource Exhaustion Considerations

If a write command is terminated as described in this sub-clause, the device server may terminate the command at any time (e.g. before processing begins, after the device server has transferred some or all of the data bytes). <**Note:** Is this already true, and not in need of repeating?>

If a write command is received and a temporary lack of physical block resources prevents the logical unit from storing the write data on the non-volatile medium, then the device server shall terminate the command with the sense key set to UNIT ATTENTION 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. <**Note:** Most existing host S/W already does a retry on UNIT ATTENTION status.>

If a write command is received and a persistent lack of physical block resources prevents the logical unit from storing the write data on the non-volatile medium, then the device server shall terminate the command with the sense key set to HARDWARE ERROR and the additional sense code set to NO ADDITIONAL SPACE CAN BE ALLOCATED. Application client recovery actions for this status are outside the scope of this standard.

If a write command is received that causes the number of available logical block resources to drop below the soft threshold value and the device server is not capable of storing the write data on the non-volatile medium, 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 PROVISIONING SOFT THRESHOLD REACHED. The recommended application client recovery action is to issue the command again.

If a write command is received that causes the number of available logical block resources to drop below the soft threshold value and the device server is capable of storing the write data on the non-volatile medium, then the device server shall perform the operation and complete the command with CHECK CONDITION status with the sense key set to RECOVERED ERROR <**Note:** One reviewer has requested to drop the distinction between success/failed operations that cross the threshold and always fail the I/O and use UNIT ATTENTION.> and the additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED. The device server shall also establish a unit attention condition for the initiator port associated with every I_T nexus other than the I_T nexus on which the command that caused the number of available logical block resources to drop below the soft threshold value was received, with the additional sense code set to PROVISIONING SOFT THRESHOLD REACHED.

If the number of available "logical block resources" drops below the soft threshold value and then increases above the soft threshold value before the PROVISIONING SOFT THRESHOLD

REACHED additional sense code is returned to the application client, the device server shall not report PROVISIONING SOFT THRESHOLD REACHED.

4.4.1.2.4 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 the specified logical blocks that have been discarded.

<...>

5. x TRIM command

5. x. 1 TRIM command overview

The TRIM command shall be implemented by device servers supporting Thin Provisioning (see 4.4.1.2). The TRIM command (see table x.1) provides information to the device server that may be used by the device server to perform device optimizations on a specified range of blocks. The method or algorithms used to perform such optimizations are not specified in this standard.

Table x.1 – TRIM 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							
7	PARAMETER LIST LENGTH							
8								
9	Control							

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

The PARAMETER LIST LENGTH field specifies the length in bytes of the TRIM 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.

<**Note:** 16 bits = FFFFh bytes or FFEh (4,094) LBA range descriptors>

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

5. x. 2 TRIM parameter list

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

Table x.3 – TRIM parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0 – 7	Reserved							
	TRIM DESCRIPTOR LIST							
8 – 23	TRIM LBA DESCRIPTOR							
…	…							
n-15 – n	TRIM LBA DESCRIPTOR							

The TRIM DESCRIPTOR LIST contains a list of LBA ranges to be operated on. The TRIM LBA DESCRIPTOR is described in table x.4. The LBAs shall be in ascending order.

Table x.4 – TRIM LBA DESCRIPTOR

Bit / Byte	7	6	5	4	3	2	1	0
0 – 3	Reserved							
4 – 11	LBA							
12 – 15	LBA COUNT							

<...>

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)							

9	LOGICAL BLOCK ADDRESS (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

Byte \ Bit	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.	

The TPE bit set to one indicates that this logical unit implements Thin Provisioning (see 4.4.1.2).

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).

<...>

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

Byte \ Bit	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (B0h)							
2	Reserved							
3	PAGE LENGTH (10h)							
4	Reserved							
5								
6	(MSB)	OPTIMAL TRANSFER LENGTH GRANULARITY						
7								(LSB)
8	(MSB)	MAXIMUM TRANSFER LENGTH						
11								(LSB)
12	(MSB)	OPTIMAL TRANSFER LENGTH						
15								(LSB)
16	(MSB)	MAXIMUM PREFETCH XDREAD XDWRITE TRANSFER LENGTH						
19								(LSB)

<...>

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. The MAXIMUM TRANSFER LENGTH field also indicates the maximum transfer length in bytes/LOGICAL BLOCK LENGTH IN BYTES (see 5.12.2) of the TRIM parameter list. <**Note:** To find the maximum transfer length in bytes of the TRIM parameter list, multiply this value by the LOGICAL BLOCK LENGTH IN BYTES. This was suggested by a host O/S team – any comments?> 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.