To: INCITS Technical Committee T10
From: Fred Knight, Network Appliance
Email: knight@netapp.com
Date: July 7, 2008
Subject: SBC-3 Thin Provisioning Commands

1.4 Revision history

Revision 0 (July 7, 2008) First revision

2.4 Related documents

spc4r15 – SCSI Primary Commands – 4
sbc3r15 – SCSI Block Commands – 3
adc3r0 – Automation/Drive Interface Commands - 3

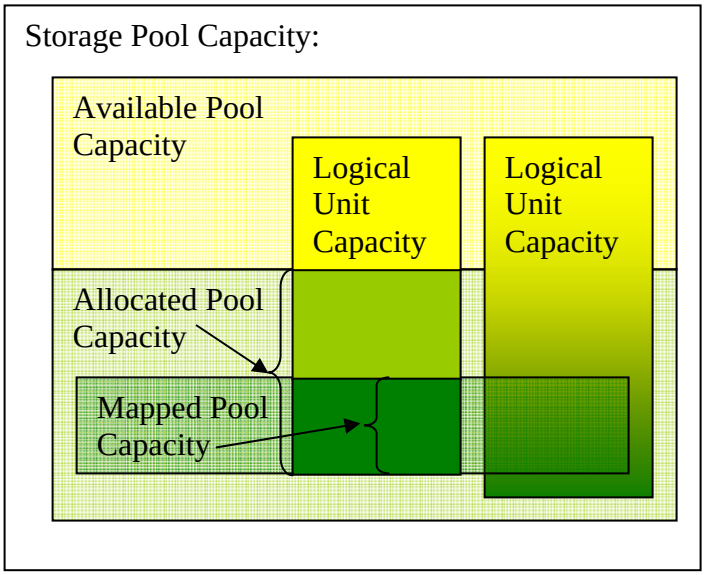
3.4 Overview

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.

As a result of the lack of a permanent mapping between physical storage and logical storage, the concept of different types of capacity is created. In addition, this creates the concepts of pools of blocks (allocated blocks, mapped blocks, available blocks, etc). This therefore requires the definition of several new terms. The proposed terms are listed in the definitions section in the body of the proposal.

This may be represented pictorially as follows:



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 would be expected to be a RECOVERED ERROR since the I/O could actually succeed. These are new types of notices to return to the application client, and as such need new ASCQ values to define these conditions.

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 it chooses (zeros, -1, etc). This "delete" function is known as a hole punch.

This proposal defines a model for operation of thinly provisioned devices, and a method for management of the provisioning of blocks for the LUN. This management method includes a method to supply a list of extent descriptors (LBA and length) to a device server, a method to query the status of particular blocks (to determine if a physically allocated block exists, or if the block is simply logical), and a method to determine and set the thin provisioning threshold points (points where errors are returned).

Proposal:

3.1.a allocated logical unit capacity: the number of logical blocks on the logical unit that are in the allocated pool.

3.1.b allocated pool: the physical blocks in the storage pool that have been allocated for use by logical units in the provisioning group (whether they are mapped or not).

3.1.c allocated pool capacity: the number of logical blocks in the allocated pool.

3.1.d available pool: the physical blocks in the storage pool that have not been allocated for use by a specific logical unit.

3.1.e available pool capacity: the number of logical blocks in the available pool.

3.1.f logical unit capacity: the total number of logical blocks available on the medium for user data and protection information (if present). The RETURNED LOGICAL BLOCK ADDRESS field of the READ CAPACITY parameter data (when the pmi bit in the READ CAPACITY command is set to zero) contains this value minus one.
<Search all of SBC for "capacity" references to see if they should change to "logical unit capacity" – Occurs 58 times>

3.1.g mapped logical block: a logical block that has a storage block mapping.

3.1.h mapped pool: the physical blocks in the allocated pool that have a storage block mapping.

3.1.i mapped pool capacity: the number of logical blocks in the mapped pool.

3.1.j mapped logical unit capacity: the number of logical blocks on the logical unit that are in the mapped pool.

3.1.k provisioning group: A group of thin provisioned logical units that share the same storage pool.

3.1.l storage block mapping: the association of a logical block with the specific physical block (or blocks) used to store the user data and protection information.

3.1.m storage pool: the pool of physical blocks used by logical units in the same provisioning group.

3.1.n storage pool capacity: the total number of logical blocks in the storage pool. This value is the available pool capacity plus the allocated pool capacity.

3.1.p unmapped logical block: a logical block that has no storage block mapping.

<...>

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 logical unit 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 logical unit 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.

4.5 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. The mechanism for determining the alignment is specified in 5.13.2 READ CAPACITY (16) (see LOWEST ALIGNED LOGICAL BLOCK ADDRESS). A mechanism for establishing the alignment is not defined by this standard.

<insert figures>

4.5.1 Full Provisioning

A device server implementing the full provisioning model ensures that a sufficient number of physical blocks are available to retain user data for all logical blocks within the logical unit capacity.

Logical Units that implement the full provisioning model shall provide sufficient physical storage to retain user data on the non-volatile medium for all blocks within the logical unit capacity.

The full provisioning model ensures that all logical blocks within the logical unit capacity are dedicated to the logical unit and are capable of storing user data written to them. The device server is capable of receiving user data transferred as a result of a write operation and storing that data on the non-volatile medium. The logical unit capacity minus one is the value returned in the RETURNED LOGICAL BLOCK ADDRESS field of the READ CAPACITY (10) parameter data when

the PMI bit is set to zero or the value returned in the LOGICAL BLOCK ADDRESS field of the READ CAPACITY (16) parameter data when the PMI bit is set to zero.

4.5.2 Thin Provisioning

A device server implementing the Thin Provisioning model may or may not ensure that a sufficient number of physical blocks are available to retain user data contents for all logical blocks within the logical unit capacity. The availability of physical blocks may change dynamically, and may not be equivalent to the logical unit capacity.

The device server may or may not be capable of receiving data transferred as a result of a write operation and storing that data on the non-volatile medium. If the device server is not capable of storing the data on the non-volatile medium, then the device server shall terminate the command with CHECK CONDITION status with the sense key set to HARD ERROR and the additional sense code set to NO ADDITIONAL SPACE CAN BE ALLOCATED.

When an application client no longer has need of the user data in a logical block (e.g., a host file system deletes a file, a database deletes a record), the application client may use the PROVISIONING MANAGEMENT command to inform the device server of this change in status of the user data in the specified range of logical blocks.

A Logical Unit that implements the thin provisioning model shall:

A. Set the TPE bit in the READ CAPACITY (16) parameter data to one;
B. Implement the PROVISIONING MODE PAGE (see xx); and
C. Implement the PROVISIONING MANAGEMENT IN/OUT Commands (see yy).

4.5.2.1 Thin Provisioning Pools

The device server manages the allocation of physical blocks through storage pools. The available pool represents those blocks in the storage pool that are available to be used (i.e. they have not been allocated for use). The allocated pool represents those blocks that have been allocated to a specific logical unit for use and may or may not be mapped to specific LBAs. The mapped pool represents those blocks that have been allocated to a specific logical unit and are mapped to a specific LBA on that logical unit. Logical Units that share blocks from the same storage pool are in the same provisioning group. Figure x.1 shows the relationship of blocks in these pools.

Logical units that implement the thin provisioning model allocate physical blocks from their provisioning group's available capacity pool to be used by the logical unit to store user data. The allocation from the available capacity pool may occur at any time (e.g., in advance of a write operation, at the time of a write operation, or when manual specified by a means not specified by this standard).

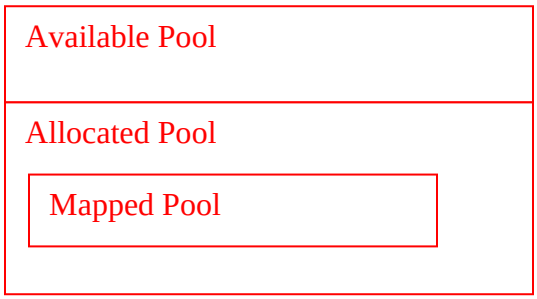


Figure x.1 – Relationship of blocks represented in various pools

4.5.3 Thin Provisioning Thresholds

Thresholds are points at which notification to the application clients occurs. The thin provisioning thresholds are defined in the PROVISIONING MODE PAGE (see xxx).

4.5.4.1 Thin Provisioning Soft Threshold

A threshold value may be set (see PROVISIONING MODE PAGE) to cause the device server to indicate to the application client when the allocated logical unit capacity reaches the specified threshold point. The Thin Provisioning Soft Threshold should be set at a point where write operations can continue to succeed, but where intervention may be needed to provide additional physical blocks to the available pool (e.g. moving physical blocks from the allocated pool to the available pool with the PROVISIONING MANAGEMENT OUT – PUNCH function, or other means outside the scope of this standard).

If a write command is received that causes the allocated logical unit capacity to exceed the soft threshold value and the device server is capable of storing the 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 and the additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED.

If a write command is received that causes the allocated logical unit capacity to exceed the soft threshold value and the device server is not capable of storing the data on the non-volatile medium (i.e., there are no more physical blocks that can be allocated to the pool of physical blocks represented by the mapped logical unit capacity), then the device server will terminate the command with CHECK CONDITION status with the sense key set to HARD ERROR and the additional sense code set to THIN PROVISIONING SOFT THRESHOLD REACHED. The device server may terminate the command at any time (e.g. before processing begins, after the device server has transferred some of the data bytes, after the device has transferred all of the data bytes).

4.5.4.2 Thin Provisioning Hard Threshold

A threshold value may be set (see PROVISIONING MODE PAGE) to cause the device server to indicate to the application client when the allocated logical unit capacity reaches the specified threshold point. The Thin Provisioning Hard Threshold should be set to a higher value than the Thin Provisioning Soft Threshold, and should represent the point where write operations are about to fail and immediate intervention may be required to provide additional physical blocks to the available pool (e.g. moving physical blocks from the allocated pool to the available pool with the PROVISIONING MANAGEMENT OUT – PUNCH function, or other means outside the scope of this standard).

If a write command is received that causes the allocated logical unit capacity to exceed the hard threshold value and the device server is capable of storing the 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 and the additional sense code set to THIN PROVISIONING HARD THRESHOLD REACHED.

If a write command is received that causes the allocated logical unit capacity to exceed the hard threshold value and the device server is not capable of storing the data on the non-volatile medium (i.e., there are no more physical blocks that can be allocated to the pool of physical blocks represented by the mapped logical unit capacity), then the device server will terminate the command with CHECK CONDITION status with the sense key set to HARD ERROR and the

additional sense code set to THIN PROVISIONING HARD THRESHOLD REACHED. The device server may terminate the command at any time (e.g. before processing begins, after the device server has transferred some of the data bytes, after the device has transferred all of the data bytes).

<...>

5. x PROVISIONING MANAGEMENT commands

The Provisioning Management commands are used to manage the provisioning state of logical units that implement the Thin Provisioning model.

5. x. 1 PROVISIONING MANAGEMENT OUT command overview

The PROVISIONING MANAGEMENT OUT command (see table x.1) requests the device server to perform a provisioning management function. The provisioning management function informs the device server to perform the specified operation on a specified range of blocks on a thinly provisioned logical unit. The method or algorithm used to perform the function is not specified in this standard. This command is mandatory for all logical units that implement the thin provisioning model (see 4.5.2).

The PROVISIONING MANAGEMENT OUT command is a service action of the SERVICE ACTION OUT command. Additional SERVICE ACTION OUT service actions are defined in ADC-2, SPC-4, and in this standard. The SERVICE ACTION OUT service actions defined only in ADC-2 apply only to logical units that return a device type of 12h in their standard INQUIRY data.

Table x.1 – PROVISIONING MANAGEMENT OUT Command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Fh)							
1	Reserved			SERVICE ACTION (nnh- 01h)				
2	Reserved							
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	Reserved							
8	Reserved							
9	Reserved							
10	(MSB) PARAMETER LIST LENGTH							
13	(LSB)							
14	PROVISIONING MANGEMENT SPECIFIC							
15	Control							

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

The Service Action field is defined in SPC-4 and shall be set to the value defined in table x.1.

The PARAMETER LIST LENGTH field specifies the length in bytes of the PROVISIONING MANAGEMENT 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.
<32 bits of length = FFFFFFFFh bytes, or 15555554h (357,913,941) LBA range descriptors – too many? 16 bits = FFFFh bytes or E37h (3,640) LBA range descriptors – enough?>

5. x .2 PROVISIONING MANAGEMENT SPECIFIC field

Code	Description	Action	Reference
01h	PUNCH	Move the specified blocks to Available pool.	5.x.2.1
02h	Erase	Overwrite user data and move the specified blocks to Available pool	5.x.2.2
03h	Security Erase	FIPS/NIST compliant destruction of user data and move the specified blocks to Available pool	5.x.2.3
04h	Pre-allocate LBAs	Move the specified blocks from Available pool to Allocated pool	5.x.2.4
05h	Map Block	Move the specified blocks from Allocated pool to Mapped pool or Available pool to Mapped pool	5.x.2.5
All others	Reserved		

5. x. 2.1 PUNCH function

The device server shall move the logical blocks specified in the PROVISIONING MANAGEMENT DESCRIPTOR LIST to the available pool. The user data and protection information contained in the blocks specified in the PROVISIONING MANAGEMENT DESCRIPTOR LIST may be discarded and replaced with vendor-specific data. More than one physical block may be prepared by each LBA descriptor. The device server shall cause vendor-specific data to be read as the user data in the specified logical blocks and shall write a default value of FFFF_FFFF_FFFF_FFFFh as the protection information for the specified logical blocks, if enabled. The data in all other logical blocks on the medium shall be preserved.

If the direct-access block device is unable to successfully complete a PUNCH command, the device server shall terminate the command with CHECK CONDITION status with the appropriate sense data (see 4.14 and SPC-4). The first LBA that can not be punched shall be returned in the COMMAND-SPECIFIC INFORMATION field of the sense data. If information about the first LBA that can not be prepared for punch is not available, or if all the LBAs have been prepared for punch, the COMMAND-SPECIFIC INFORMATION field shall be set to FFFF_FFFFh if fixed format sense data is being used or FFFF_FFFF_FFFF_FFFFh if descriptor format sense data is being used.

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

NOTE 1 - If the PUNCH command returns CHECK CONDITION status and the sense data COMMAND-SPECIFIC INFORMATION field contains a valid LBA, the application client should remove all LBAs represented in the PUNCH PARAMETER list prior to the one returned in the COMMAND-SPECIFIC INFORMATION field. The application client should perform any corrective action indicated by the sense data and then reissue the PUNCH command with the new PUNCH PARAMETER list.

5. x. 2.2 ERASE function

The device server shall destroy the contents of the user data area of the logical blocks specified in the PROVISIONING MANAGEMENT DESCRIPTOR LIST and then perform the PUNCH function.

Add error cases

5. x. 2.3 SECURITY ERASE function

The device server shall destroy the contents of the user data area of the logical blocks specified in the PROVISIONING MANAGEMENT DESCRIPTOR LIST using a FIPS/NIST compliant method and then perform the PUNCH function.

Add error cases

5. x. 2.4 PRE-ALLOCATE function

The device server shall allocate sufficient blocks from the available pool to the allocated pool for the logical unit to map the logical blocks specified in the PROVISIONING MANAGEMENT DESCRIPTOR LIST.

If sufficient blocks can not be allocated…<add error cases>

5. x. 2.5 MAP BLOCK function

The device server shall establish the mapping from blocks in the allocated pool to the logical blocks specified in the PROVISIONING MANAGEMENT DESCRIPTOR LIST. If insufficient blocks are in the allocated pool, then additional blocks shall be used from the available pool.

If sufficient blocks can not be mapped…<add error cases>

5. x. 3 PROVISIONING MANAGEMENT parameter list

The PROVISIONING MANAGEMENT parameter list (see table x.2) contains an eight-byte parameter list header followed by a PROVISIONING MANAGEMENT DESCRIPTOR LIST containing one or more LBA DESCRIPTORs.

Table x.2 – PROVISIONING MANAGEMENT PARAMETER LIST

Bit / Byte	7	6	5	4	3	2	1	0
0	PROVISIONING MANAGEMENT PARAMETER LIST LENGTH (n-3)							
3								
4	Reserved							
7								
PROVISIONING MANAGEMENT DESCRIPTOR LIST								
11	LBA DESCRIPTOR							
19								

…	…
n-11	LBA DESCRIPTOR
n	

The PROVISIONING MANAGEMENT PARAMETER LIST LENGTH field indicates the number of bytes contained in the PROVISIONING MANAGEMENT PARAMETER LIST.

The PROVISIONING MANAGEMENT DESCRIPTOR LIST contains a list of LBA ranges to be operated on. Each LBA DESCRIPTOR shall contain an eight-byte LBA followed by a four-byte count (table x.3). The LBA DESCRIPTOR shall be in listed in ascending order.

Table x.3 – PROVISIONING MANAGEMENT LBA DESCRIPTOR

Bit / Byte	7	6	5	4	3	2	1	0
0	LBA							
7								
8	LBA COUNT							
11								

5. x. 4 PROVISIONING MANAGEMENT IN command overview

The PROVISIONING MANAGEMENT IN command (see table x.4) requests that the device server send provisioning management information to the application client. This command is mandatory for all logical units that implement the thin provisioning model (see 4.5.2).

The PROVISIONING MANAGEMENT IN command is a service action of the SERVICE ACTION IN command. Additional SERVICE ACTION IN service actions are defined in ADC-2, SPC-4, and in this standard. The SERVICE ACTION IN service actions defined only in ADC-2 apply only to logical units that return a device type of 12h in their standard INQUIRY data.

This command returns the status of a specified LBA to the data in buffer. That data contains a mask of status bits per LBA to specify the provisioning status of that LBA. The first byte in the returned buffer contains the status for the specified LBA. The second byte specifies the status for the specified LBA + 1. etc… until the specified buffer length has been returned. <Must translate this clause to spec-eze.>

Table x.4 – PROVISIONING MANAGEMENT IN Command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (nnh- 01h)				
2	(MSB) LOGICAL BLOCK ADDRESS							
9								(LSB)
10	(MSB) ALLOCATION LENGTH							
13								(LSB)
14	PROVISIONING MANGEMENT SPECIFIC (0h)							

15	Control

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

The Service Action field is defined in SPC-4 and shall be set to the value defined in table x.4.

The LOGICAL BLOCK ADDRESS field specifies the LBA of the first logical block (see 4.4) accessed by this command. If the specified LBA exceeds the logical unit 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 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 PROVISIONING MANAGEMENT SPECIFIC field shall be set to zero.

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

5. x. 4.1 PROVISIONING MANAGEMENT IN parameter data

The PROVISIONING MANAGEMENT IN PARAMETER DATA (see table x.5) contains an eight-byte header followed by one or more LBA STATUS DESCRIPTORS.

Table x.5 PROVISIONING MANAGEMENT parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	PROVISIONING MANAGEMENT IN PARAMETER LIST LENGTH (n-3)							
3								
4	Reserved							
7								
LBA STATUS DESCRIPTOR LIST								
8	LBA STATUS DESCRIPTOR (LBA)							
9	LBA STATUS DESCRIPTOR (LBA + 1)							
10	LBA STATUS DESCRIPTOR (LBA + 2)							
...	...							
n	LBA STATUS DESCRIPTOR (LBA + n+8)							

5. x. 4.1.1 LBA STATUS DESCRIPTOR

The LBA STATUS DESCRIPTOR (see table x.6) contains LBA provisioning status information.

Table x.6 LBA STATUS DESCRIPTOR

Bit / Byte	7	6	5	4	3	2	1	0

	Reserved	MAP

A MAP bit set to one indicates that the specified LBA is mapped to a physical block.

A MAP bit set to one indicates that the specified LBA is not mapped to a physical block.

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	TPE	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 the Thin Provisioning (see 4.5.2) model.

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).

Still to do's:
Verify REPORT SUPPORTED OPERATION CODES works with these commands and service actions. What about the pseudo-sub-service actions? The Provisioning Management Specific field for both IN and OUT?

Add methods for host to find out which sub-functions are supported, which capabilities are supported. Consider De-Duplication technologies and ATA translation – Data Set Mgmt Trim command.

6.3.8 PROVISIONING MODE PAGE – lots more words needed here

The Provisioning mode page (see table x.7) defines the parameters that effect thin provisioning.

MODE SENSE – read all fields

MODE SELECT – set soft and hard threshold values

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (1b)	PAGE CODE (xxh)					
1	SUBPAGE CODE (yyh)							
2	(MSB)							
3	PAGE LENGTH (n-3) (LSB)							
4-7	Reserved							
8-15	AVAILABLE POOL CAPACITY							
16-23	ALLOCATED POOL CAPACITY							
24-31	LOGICAL UNIT CAPACITY - duplicate here? - NO							
32-39	LOGICAL UNIT ALLOCATED CAPACITY							
40-47	LOGICAL UNIT MAPPED CAPACITY							
48-49	PROVISIONING GROUP ID							
50	SOFT THRESHOLD PERCENT							
51	HARD THRESHOLD PERCENT							
52-95	Reserved							

<Make consistent – usage of “logical unit allocated capacity” and “allocated logical unit capacity” – also “logical unit mapped capacity” and “mapped logical unit capacity”>

Anything interesting related to snapshots and reporting (or not) of their capacity usage?

Make sure all these commands are covered by the text in the model clause:

5.3 ORWRITE command

5.26 WRITE (6) command

5.27 WRITE (10) command

5.28 WRITE (12) command

5.29 WRITE (16) command

5.30 WRITE (32) command

5.31 WRITE AND VERIFY (10) command

5.32 WRITE AND VERIFY (12) command

5.33 WRITE AND VERIFY (16) command

5.34 WRITE AND VERIFY (32) command

5.35 WRITE LONG (10) command

5.36 WRITE LONG (16) command

5.37 WRITE SAME (10) command

5.38 WRITE SAME (16) command

5.39 WRITE SAME (32) command

5.42 XDWRITE (10) command

5.43 XDWRITE (32) command

5.44 XDWRITEREAD (10) command

5.45 XDWRITEREAD (32) command

5.46 XPWRITE (10) command

5.47 XPWRITE (32) command