To: INCITS Technical Committee T10
From: Fred Knight, NetApp
David Black, EMC
Email: knight@netapp.com
black_david@emc.com
Date: May 19, 2009
Subject: SBC-3 Thin Provisioning Management Commands

1) *Revision history*

Revision 0 (Sept 9, 2008) First revision (r0)
Revision 1 (Nov 4, 2008) Make adjustments based on changes to 08-149
Revision 2 (Jan 7, 2009) Add additional management capabilities
(Apr 30, 2009) Change command name to GET LBA STATUS and add ALLOC bit.
Revision 3 (May 19, 2009) Remove ALLOC state and cover just MAPPED and UNMAPPED states

2) *Related documents*

spc4r18 – SCSI Primary Commands – 4
sbc3r18 – SCSI Block Commands – 3

3) *Overview*

The addition of thin provisioning to SBC-3 (cf. 08-365r5) defined mapped and unmapped LBA states, but did not provide any way for an initiator to determine whether LBAs are mapped vs. unmapped. This proposal adds that functionality in the form of a general command that returns status information, ~~including~~ starting with the mapped vs. unmapped status of LBAs.

One important use of the mapped vs. unmapped status information involves copying a thinly provisioned logical unit. An initiator may optimize such a copy by not copying the unmapped LBAs because no user data is stored in them, but to employ this optimization, the initiator needs to know which LBAs are unmapped.

This command may be expanded in the future to report other LBA characteristics.

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

Proposal:

Table 3 – SBC Commands that are allowed in the presence of various reservations

Command	WE	EA	RR	WE-RR	EA-RR
GET LBA STATUS	Allow	Conflict	Allow	Allow	Conflict

<…>

Table 13 – Associations between commands and CbCS permissions

Command	DATA READ	DATA WRITE	PARM READ	PARAM WRITE	PHY ACC
GET LBA STATUS			1		

<…>

Table 14 – Commands for direct-access block devices

Command name	OP code	Type	PI	Reference
GET LBA STATUS	9Eh/12h	O	no	5.x.1

<Editor note: Op code is up to the editor.>

<…>

5. x GET LBA STATUS command

5. x. 1 GET LBA STATUS command overview

The GET LBA STATUS command should be implemented by device servers supporting thin provisioning (see 4.6.3.1). The GET LBA STATUS command (see table x.1) requests that the device server send status information for the specified logical block addresses to the application client.

Table x.1 – Provisioning Management Command

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (nnh- 12h)				
2	(MSB)							
9	STARTING LOGICAL BLOCK ADDRESS							(LSB)
10	(MSB)							
13	ALLOCATION LENGTH							(LSB)
14	RESERVED							
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.4.

<Should this be a service action, or a new dedicated command?>

The STARTING LOGICAL BLOCK ADDRESS field specifies the LBA of the first logical block addressed by this command. If the specified LBA exceeds the capacity of the medium (see 4.4), 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.

<Should (see 4.4) be added to "capacity of the medium" statements all over SBC? Or is this a well known concept in light of the "LOGICAL BLOCK ADDRESS OUT OF RANGE" statement.>

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 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.

<Authors note – the above text now matches READ CAPACITY (16) use of the allocation length field as suggested, and includes below, a suggested allocation length similar to REPORT LUNS.>

The ~~application client should specify an ALLOCATION LENGTH that is~~allocation length should be (a multiple of 16) + 8. ~~The device server shall terminate transfers to the data-in buffer when:~~

~~a)the number of bytes specified by the ALLOCATION LENGTH field has been transferred,~~
~~b)when data representing all logical blocks higher than the specified logical block address has been transferred, or~~
~~c)the device server has transferred at least 1 LBA descriptor.~~

~~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. x. 2 GET LBA STATUS parameter data

5. x. 2.1 GET LBA STATUS parameter Overview

The GET LBA STATUS parameter data (see table x.2) contains an eight-byte header followed by one or more LBA status descriptors.

Table x.2 GET LBA STATUS parameter data

Bit / Byte	7	6	5	4	3	2	1	0
0	PARAMETER DATA LENGTH (n-3)							
3								
4	Reserved							
7								
LBA STATUS DESCRIPTOR LIST								
8-23	LBA STATUS DESCRIPTOR 1							
...	...							
n	LBA STATUS DESCRIPTOR m							

The STATUS LIST LENGTH field indicates the length in bytes of the parameter data. The relationship between the STATUS LIST LENGTH field and the ALLOCATION LENGTH field in the CDB is defined in SPC-4.

Note X: LBA provisioning status changes may occur during or after processing of the GET LBA STATUS command.

5. x. 2.1.1 LBA status descriptor

The LBA status descriptor (see table x.3) contains LBA status information for one or more LBAs.

Table x.3 LBA status descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB) STARTING LOGICAL BLOCK ADDRESS							
7								(LSB)
8	(MSB) NUMBER OF LOGICAL BLOCKS							
11								(LSB)
12	RESERVED				~~LBA~~ PROVISIONING STATUS			
13	RESERVED							
15								

The STARTING LOGICAL BLOCK ADDRESS field contains the starting LBA of the LBA extent for which this descriptor reports LBA status. The NUMBER OF LOGICAL BLOCKS field contains the number of logical blocks in that extent.

The STARTING LOGICAL BLOCK ADDRESS field in the first LBA status descriptor returned by the GET LBA STATUS command shall be the LBA supplied in the STARTING LOGICAL BLOCK ADDRESS field of the CDB. For subsequent LBA status descriptors, the contents of the STARTING LOGICAL BLOCK ADDRESS field shall be the sum of the contents of:

a) the STARTING LOGICAL BLOCK ADDRESS field in the previous LBA status descriptor; and
b) the NUMBER OF LOGICAL BLOCKS field in the previous LBA status descriptor.

~~A MAP bit set to one indicates that the specified LBA range is:~~

~~a)mapped to a physical block range; and~~
~~b)contains persistent user data and protection information if enabled.~~

~~A MAP bit set to zero indicates that the specified LBA range is:~~

~~a)not mapped to a physical block range; and~~
~~b)does not contain persistent user data or protection information if enabled.~~

The PROVISIONING STATUS field is described in table X.4

Table X.4 – LBA status field

Name	Code	Description
Undefined	0h	The status of the LBA is unknown
MAPPED	1h	The LBA is mapped (see 4.6.4.2)
UNMAPPED	2h	The LBA is unmapped (see 4.6.4.3)
All others		Reserved

If the logical unit is fully provisioned, then the PROVISIONING STATUS for all LBAs shall be mapped (see 4.6.2).

Adjacent LBA status descriptors should have different values for the PROVISIONING STATUS field.

If the last LBA status descriptor in the GET LBA STATUS parameter data specifies an LBA extend that does not include the last LBA for the logical unit, then the next LBA may have the same provisioning status as that LBA extent.