To: INCITS Technical Committee T10
From: David L. Black, EMC
Email: black_david@emc.com
Date: Apr 29, 2009
Subject: SBC-3: GET LBA STATUS command

1) *Revision history*

Revision 0 (Sept 9, 2008) First revision (r0)
Revision 1 (Apr 29, 2009) Generalize from thin provisioning management to a general GET LBA STATUS command that can return other sorts of per-LBA status. Add ALLOC bit to returned LBA descriptors. (r1)
Revision 2 (May 6, 2009) Temporary author change, content is identical to previous version. (r2)

2) *Related documents*

sbc3r18 – SCSI Block Commands – 3
08-356r5 – Thin Provisioning Commands
T13/e07154r6 – ATA8-ACS2 accepted TRIM proposal
T13/e08137r4 – ATA8-ACS2 accepted Deterministic read after TRIM proposal

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 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.

In addition, it is also useful to report whether a write operation consumes physical block resources and hence can fail due to a space allocation failure. This is not the same as reporting whether an LBA is mapped because:

- A device server implementation may reserve space for specific unmapped LBAs in order to ensure that writes to those LBAs cannot fail for lack of physical block resources.

- A device server implementation may manage physical block resources in a fashion that causes writes to some mapped blocks to consume such resources and hence be capable of causing space allocation failures.

This information may help initiators avoid space allocation failures when writing to LBAs for which write failures are particularly problematic to an initiator.

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:

5.x GET LBA STATUS command

5.x.1 GET LBA STATUS command overview

The GET LBA STATUS command shall be implemented by device servers that support thin provisioning (see 4.4.1.2). 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 – GET LBA STATUS Command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (9Eh)							
1	Reserved			SERVICE ACTION (nnh)				
2	(MSB)			LOGICAL BLOCK ADDRESS				
9								(LSB)
10	(MSB)			ALLOCATION LENGTH				
13								(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.1.

Editor's Note: This is currently specified as a service action. Consider whether to use a dedicated operation code.

The 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, 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.

Editor's Note: "capacity of the medium" is the usual SBC-3 wording for this text. It may not be correct for thinly provisioned logical units, in which case a global substitution is needed.

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 application client should specify an ALLOCATION LENGTH that is (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 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

Byte \ Bit	7	6	5	4	3	2	1	0
0 – 3	STATUS LIST LENGTH (n-3)							
4	Reserved							
5 – 7	Reserved							
	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.

Editor's Note: The 1DESC bit that was initially placed here will be moved to a VPD page. That bit has the following function: A 1DESC bit set to one indicates that the device server will only report one LBA status descriptor. A 1DESC bit set to zero indicates that the device server may report more than one LBA status descriptor.

5.x.2.2 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

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	LOGICAL BLOCK ADDRESS						
7								(LSB)
8	(MSB)	NUMBER OF LOGICAL BLOCKS						
11								(LSB)
12	RESERVED						ALLOC	MAP
13	RESERVED							
14	RESERVED							
15	RESERVED							

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

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

An ALLOC bit set to one indicates that write operations to the specified LBAs do not cause the two space allocation failures specified in 4.6.3.4.

An ALLOC bit set to zero indicates that write operations to the specified LBAs may cause either of the two space allocation failures specified in 4.6.3.4.

For a device server that implements no thin provisioning functionality beyond that specified in 4.6 (including meeting the command requirements in 4.6.3.1), the values of the MAP and ALLOC bits are always identical. A device server may support vendor-specific physical block allocation functionality that causes these bits to have different values.

Editor's Note: The purpose of the ALLOC bit is to report that space has been allocated for these specific LBAs and hence writes can never fail due to resource shortages. It may be better to describe the concept of allocation in the model clause and refer to that concept from here. Does the ALLOC bit need to be a 2 bit field with "yes", "no" and "not reported" values?

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.