To: INCITS Technical Committee T10
From: David L. Black, EMC
Email: black_david@emc.com
Date: December 10, 2008
Subject: SBC-3: WRITE SAME unmap bit (08-356r4)

1) Revision history

Revision 0 (Sep 4, 2008) First revision (r0)

Revision 1 (Oct 27, 2008) Change name of bit to avoid confusion with ATA TRIM command.

Revision 2 (Oct 31, 2008) Revised based on comments from Rob Elliott. The bit now has the obvious name of UNMAP, and its meaning is "should unmap", but the device server decides which blocks are unmapped.

Revision 3 (Nov 6, 2008) Align with latest version of 08-149 proposal.

Revision 4 (Dec 10, 2008) Alignment with 08-149r7. Remove changes to WRITE SAME (10).

2) Related documents

sbc3r16 – SCSI Block Commands – 3

08-149r7 – SBC-3 Thin Provisioning Commands

3) Overview

08-149r5 proposes a new SCSI UNMAP command that enables a device server to unmap physical blocks for logical blocks whose contents are no longer needed. The read behavior of logical blocks affected by the UNMAP command is non deterministic by design; the device server chooses what (if any) physical blocks to unmap and chooses what value to report when a READ command is processed for a logical block whose physical block has been unmapped. The resulting flexibility is very useful to implementations, and fits situations where the value of the data read from an unmapped logical block is not important. A disk or file system defragmenter that unmaps blocks in the continguous free area resulting from defragmentation is an example because the defragmenter does not care what the results of reading the freed blocks are and the filesystem does not rely on the contents of the blocks in the free area.

There are also situations where the value of the data read from an unmapped block is important. Filesystems provide a class of examples because:

- It is useful for filesystems to allow unmapping of physical blocks for logical blocks that are not going to be used for a significant period of time.
- When those logical blocks are put back into use (e.g., in response to increased space demand), the blocks may have to be initialized to zeroes.

Many filesystems are required to initialize some blocks being placed into use (e.g., indirect blocks, or a block whose first access does not overwrite the entire block). If the SCSI operation that allows the device server to unmap physical blocks also initializes all of the affected logical blocks, those logical blocks do not have to be re-initialized in order to subsequently return them to use. This has the potential to shift logical block initialization operations from high demand times (when blocks that have not been used for some time are being returned to use) towards lower demand times when I/O resources may be more readily available.

This document proposes adding a bit to the WRITE SAME commands for the deterministic class of use cases exemplified by filesystems. When the data written by WRITE SAME matches the data pattern that results from reading an unmapped logical block, it may be useful for a device server to unmap the logical blocks. This proposal adds an UNMAP bit to the CDB to indicate that the device server may unmap any or all of the logical blocks, but does not require that any or all of the logical blocks be unmapped. The UNMAP bit should only be set by an application client that does not anticipate reuse (e.g., writes) in the near future, as a write to an unmapped block remaps it.

No changes are made to WRITE SAME (10). Thin provisioning requires support of READ CAPACITY (16) (see 08-149r6), hence one can expect 16-byte CDBs to be supported by initiators that are aware of thin provisioning

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

Proposal:

5.37 WRITE SAME (10) command

No changes to WRITE SAME (10).

5.38 WRITE SAME (16) command

Add an UNMAP bit to the CDB:

Table 90 – WRITE SAME (16) Command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (93h)							
1	WRPROTECT			**Reserved**	**~~Reserved~~ UNMAP**	PBDATA	LBDATA	Reserved
2	(MSB) LOGICAL BLOCK ADDRESS							
9								(LSB)
10	(MSB) NUMBER OF LOGICAL BLOCKS							
13								(LSB)
14	Reserved				GROUP NUMBER			
15	CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 90.

If the logical unit is thin provisioned (see 4.4.1.5), then an UNMAP bit set to one requests that for each specified LBA:

a) A mapped LBA should be unmapped, or may remain mapped; and
b) An unmapped LBA should remain unmapped, or may be mapped.

If:

a) The logical unit is thin provisioned;
b) At least one bit in the data out buffer for this command is not zero; and
c) The TPRZ bit is set to one in the parameter data returned by READ CAPACITY (16) (see 5.13.2).

Then the UNMAP bit shall be ignored, and this command shall not unmap any LBA.

If the logical unit not thin provisioned or the UNMAP bit is set to zero, then this command shall not unmap any LBA.

For each specified LBA, the user data retrieved by the next read operation or verify operation shall be the same as the user data contents of the logical block transferred from the data-out buffer by processing of this command.

Note: The device server may map any specified LBA in order to meet this requirement.

See the WRITE SAME (10) command (see 5.37) for the definitions of the other fields in this command.

5.38 WRITE SAME (32) command

Add an UNMAP bit to the CDB:

Table 91 – WRITE SAME (32) Command

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2 5	Reserved							
6	Reserved			GROUP NUMBER				
7	ADDITIONAL CDB LENGTH (18h)							
8 9	(MSB) SERVICE ACTION (000Dh)							(LSB)
10	WRPROTECT			**Reserved**	**~~Reserved~~ UNMAP**	PBDATA	LBDATA	Obsolete
11	Reserved							
12 31	**No change to remainder of CDB fields**							

The OPERATION CODE field, ADDITIONAL CDB LENGTH field, and SERVICE ACTION field are defined in SPC-4 and shall be set to the values defined in table 91.

See the WRITE SAME (10) command (see 5.37) for the definitions of the CONTROL byte, GROUP NUMBER field, the WRPROTECT field, the PBDATA bit, the LBDATA bit, the LOGICAL BLOCK ADDRESS field, and the NUMBER OF LOGICAL BLOCKS field. See the WRITE SAME (16) command (see 5.38) for the definition of the UNMAP bit.