EMC²
where information lives®

To: INCITS Technical Committee T10
From: David L. Black, EMC
Email: black_david@emc.com
Date: October 31, 2008
Subject: SBC-3: WRITE SAME unmap bit (08-356r2)

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 and the SCSI PUNCH proposal, as these have different behaviors.

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 if able", but the device server decides which blocks are unmapped.

2) *Related documents*

sbc3r16 – SCSI Block Commands – 3

08-149r3 – SBC-3 Thin Provisioning Commands

3) *Overview*

08-149r3 proposes a new SCSI PUNCH 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 PUNCH 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 MUNM (May 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 MUNM 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.

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

Proposal:

In the new section 4.4.1.3.1 "Transition from Mapped to Unmapped" added by 08-149r3, change the contents of the section to read:

This transition may occur as a consequence of successful execution of a PUNCH command or of a WRITE SAME command with the UNMAP bit set (see 4.4.1.5.2).

5.37 WRITE SAME (10) command

Add an UNMAP bit to the CDB:

Table 88 – WRITE SAME (10) Command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (41h)							
1	WRPROTECT			**Reserved**	~~**Reserved**~~ **UNMAP**	PBDATA	LBDATA	Obsolete
2	(MSB) LOGICAL BLOCK ADDRESS							
5								(LSB)
6	Reserved			GROUP NUMBER				
7	(MSB) NUMBER OF LOGICAL BLOCKS							
8								(LSB)
9	CONTROL							

The OPERATION CODE field is defined in SPC-4 and shall be set to the value defined in table 88.

See the WRITE (10) command (see 5.27) for the definitions of the CONTROL byte and WRPROTECT field. See the PRE-FETCH (10) command (see 5.4) for the definition of the LOGICAL BLOCK ADDRESS field. See the PRE-FETCH (10) command (see 5.4) and 4.18 for the definition of the GROUP NUMBER field.

If the logical unit is thin provisioned (see 4.4.x), and the contents of the data-out buffer are either data with all bits set to zero or data with all bits set to one, then an UNMAP bit set to one specifies that the device server should unmap the logical blocks designated by the WRITE SAME command. If the device server is unable to unmap all of the designated logical blocks, the device server should unmap the logical blocks that it is able to unmap. The protection information for any unmapped logical block shall be FFFF_FFFF_FFFF_FFFF. An UNMAP bit set to zero shall have no effect.

Note: May need text here to indicate that the "should" allows unmapping to not occur when the data pattern doesn't match the device's data pattern for read of unmapped blocks. This would be simpler if there was only one unmapped data pattern (all zeros).

When the UNMAP bit is set to one, it shall not be an error for any of the logical blocks designated by the WRITE SAME command to be in the unmapped state when device server processing of the WRITE SAME command commences, and it shall not an error for any of the logical blocks designated by the WRITE SAME command to be in the mapped state when device server processing of the WRITE SAME command completes.

The UNMAP bit shall be ignored if the logical unit is not thin provisioned or if the contents of the data-out buffer are other than data with all bits set to zero or data with all bits set to one.

The value of the UNMAP bit shall not affect the data obtained by subsequent commands that read or otherwise use the data for the logical blocks; the data used by such commands shall be the same as if the UNMAP bit was ignored in the WRITE SAME command.

Note: The previous paragraph may be redundant with text elsewhere. The rule is important, but may be appropriate to state somewhere else.

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.

See the WRITE SAME (10) command (see 5.37) for the definitions of the other fields in this command. <No change is needed to this text.>

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	Reserved							
5								
6	Reserved			GROUP NUMBER				
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)							
9	SERVICE ACTION (000Dh)							(LSB)
10	WRPROTECT			**Reserved**	**~~Reserved~~ UNMAP**	PBDATA	LBDATA	Obsolete
11	Reserved							
12	**No change to remainder of CDB fields**							
31								

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 UNMAP bit, the PBDATA bit, the LBDATA bit, the LOGICAL BLOCK ADDRESS field, and the NUMBER OF LOGICAL BLOCKS field.