| | |
|---|---|
| **Date:** | October 24, 2006 |
| **To:** | T10 Committee (SCSI) |
| **From:** | Roger Cummings (Symantec) |
| **Subject:** | T10/06-393r1 On-disk bitmap support |

## Revision History

| | | |
|---|---|---|
| 06-393r0 | (August 31, 2006) | Original |
| 06-393r1 | (October 24, 2006) | Modified in response Feedback from September 2006 CAP meeting. Replaced WRPROTECT field by ORPROTECT field, added details (provided by George Penokie) of command operation wrt the Protection Information model. Added changes to Protection Information model section. |

## Background

Distributed applications are making increasing use of bitmap structures to track state and control the operation of various of their features. A number of these bitmap structures are located on disk storage in order that they can survive the failure of one or more servers. While some of these bitmaps are related to the storage on which they reside (e.g. a 'dirty" sector list or a mirror resynchronization scoreboard), many are related to general application features such as:

- Cluster membership
- Cache Status
- Database table manipulation
- Parallelized processing of large data sets

In order to change the status of a single bit in such a bitmap, an application has to read all or a portion of the bitmap structure, change the status of the desired bit, and write the result back to the storage. However if during that process another Initiator begins a similar process to update a different bit, the state of the bitmap can become corrupted.

It is therefore necessary to establish at least "write exclusive" access to the Logical Unit containing the bitmap for the entire duration of the bitmap update process. This represents a significant impediment to the operation of the distributed application, given that by definition it relies on shared access to storage.

Write exclusive access can either be established within the distributed application by the use of global locks and/or a designated bitmap manager, or by the use of SCSI Reservations or Persistent Reservations. The former creates a bottleneck and single point of failure within the application, the latter adds significant overhead to the storage access as follows:

Consider the case where a number of servers running a cluster application are accessing a set of shared storage devices. In normal operation, each server Registers with each device, and an All Registrants SCSI Persistent Reservation is established with each device. However when a server needs to update a bit in a bitmap hosted on one of the shared storage devices it has to:

1) Preempt the existing All Registrants Persistent Reservation, and establish a Write Exclusive Persistent Reservation;
2) Read all or part of the bitmap structure from the storage;
3) Change the state of one or more bits in the bitmap structure;
4) Write all or part of the bitmap structure to the storage;
5) Preempt the existing Write Exclusive Persistent Reservation, and re-establish an All Registrants Persistent Reservation.

During the above process, none of the other servers running the cluster application will be able to access other data on the same storage device as the bitmap structure. After completion of the above process, all of the other servers running the cluster application will also need to reRegister with that storage device.

Again, many thanks to George Penokie for providing the tables on checking protection information.

## Proposal

The need to establish write exclusive access during a bitmap update could be avoided if a single SCSI command was capable of changing the state of one or more bits in the bitmap structure regardless of the setting of the rest of the structure. This would allow the "read-update-write" cycle to take place within the Device Server as an uninterrupted series of actions. This document proposes the definition of a new ORWRITE command for SBC-3 to perform this function.

Note that it is important that this new command only be able to set a bit in the bitmap to a one, to ensure that the final state of the bitmap is independent of the order in which the commands against it are processed. This is the reason that a new command is needed, as none of the existing X?WRITE commands will serve the same purpose.

## Suggested Changes

Add the following row to Table 3

**Table 3 — SBC-2 commands that are allowed in the presence of various reservations**

| Command | Addressed logical unit has this type of persistent reservation held by another I_T nexus | | | | |
|---|---|---|---|---|---|
| | From any I_T nexus | | From registered I_T nexus (RR all types) | From I_T nexus not registered | |
| | Write Exclusive | Exclusive Access | | Write Exclusive - RR | Exclusive Access - RR |
| ORWRITE | Conflict | Conflict | Allowed | Conflict | Conflict |
| **Key:** RR = Registrants Only or All Registrants<br>**Allowed:** Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present should complete normally.<br>**Conflict:** Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present shall not be performed and the device server shall terminate the command with RESERVATION CONFLICT status. | | | | | |
| [a] Logical units claiming compliance with previous versions of this standard (e.g., SBC-2) may return RESERVATION CONFLICT in this case. Logical units may report whether certain commands are allowed in the PERSISTENT RESERVE IN command REPORT CAPABILITIES service action parameter data ALLOW COMMANDS field (see SPC-4). | | | | | |

Amend 4.17.2 as follows:

## 4.17.2 Protection types

### 4.17.2.1 Protection types overview

The content of protection information is dependent on the type of protection to which a logical unit has been formatted.

The type of protection supported by the logical unit shall be indicated in the SPT field in the Extended INQUIRY Data VPD page (see SPC-4). The current protection type shall be indicated in the P_TYPE field in the READ CAPACITY(16) command (see 5.12).

An application client may format the logical unit to a specific type of protection using the RTO_REQ bit and PROTECTION FIELD USAGE field in the FORMAT UNIT command (see 5.2).

The media access commands are processed in a different manner by a device server depending on the type of protection in effect. When used in relation to types of protection, the term “media access commands” is defined as the following commands:

a) READ (10);
b) READ (12);
c) READ (16);
d) READ (32);
e) VERIFY (10);
f) VERIFY (12);
g) VERIFY (16);
h) VERIFY (32);
i) WRITE (10);
j) WRITE (12);
k) WRITE (16);
l) WRITE (32);
m) WRITE AND VERIFY (10);
n) WRITE AND VERIFY (12);
o) WRITE AND VERIFY (16);
p) WRITE AND VERIFY (32);
q) WRITE SAME (10);
r) WRITE SAME (16);
s) WRITE SAME (32);
t) XDWRITE (10);
u) XDWRITE (32);
v) XDWRITEREAD (10);
w) XDWRITEREAD (32);
x) XPWRITE (10);
y) XPWRITE (32);
z) XDREAD (10);
aa) XDREAD (32); and
ab) ORWRITE.

The device server may allow the READ (6) command (see 5.6) and the WRITE (6) command (see 5.25) regardless of the type of protection to which the logical unit has been formatted.

### 4.17.2.2 Type 0 protection

Type 0 protection defines no protection over that which is defined within the transport protocol.

A logical unit that has been formatted with protection information disabled (see 5.2) or a logical unit that does not support protection information (i.e., the PROTECT bit set to zero in the Standard INQUIRY data (see SPC-4)) has type 0 protection.

If type 0 protection is enabled and the RDPROTECT field, WRPROTECT field, ~~or~~ VRPROTECT, or ORPROTECT field is set to a non-zero value, then media commands are invalid and may be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE.

If type 0 protection is enabled and the RDPROTECT field, WRPROTECT field, ~~or~~ VRPROTECT, or ORPROTECT field is set to a zero value, then the following media commands are invalid and shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

a) READ (32);
b) VERIFY (32);
c) WRITE (32);
d) WRITE AND VERIFY (32); and
e) WRITE SAME (32).

### 4.17.2.3 Type 1 protection

Type 1 protection:

a) defines the content of the LOGICAL BLOCK GUARD field;
b) does not define the content of the LOGICAL BLOCK APPLICATION TAG field; and
c) defines the content the LOGICAL BLOCK REFERENCE TAG field.

If type 1 protection is enabled, then the following media commands are invalid and shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

a) READ (32);
b) VERIFY (32);
c) WRITE (32);
d) WRITE AND VERIFY (32); and
e) WRITE SAME (32).

For valid media access commands in which the RDPROTECT field, WRPROTECT field, ~~or~~ VRPROTECT, or ORPROTECT field is set to:

a) zero, the data-in buffer and/or data-out buffer associated with those commands shall consist of logical blocks with only user data; or
b) a non-zero value, the data-in buffer and/or data-out buffer shall consist of logical blocks with both user data and protection information.

### 4.17.2.4 Type 2 protection

Type 2 protection:

a) defines the content of the LOGICAL BLOCK GUARD field;
b) does not define the content of the LOGICAL BLOCK APPLICATION TAG field; and
c) defines, except for the first logical block addressed by the command, the content of the LOGICAL BLOCK REFERENCE TAG field.

If type 2 protection is enabled and the RDPROTECT field, WRPROTECT field, ~~or~~ VRPROTECT, or ORPROTECT field is set to a non-zero value, then the following media commands are invalid and shall be terminated with CHECK

CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

a) READ (10);
b) READ (12);
c) READ (16);
d) VERIFY (10);
e) VERIFY (12);
f) VERIFY (16);
g) WRITE (10);
h) WRITE (12);
i) WRITE (16);
j) WRITE AND VERIFY (10);
k) WRITE AND VERIFY (12);
l) WRITE AND VERIFY (16);
m) WRITE SAME (10);
n) WRITE SAME (16);
o) XDWRITE (10);
p) XDWRITE (32);
q) XDWRITEREAD (10);
r) XDWRITEREAD (32);
s) XPWRITE (10);
t) XPWRITE (32);
u) XDREAD (10);
v) XDREAD (32); and
w) ORWRITE.

For valid media access commands in which the RDPROTECT field, WRPROTECT field, ~~or~~ VRPROTECT, or ORPROTECT field is set to:

a) zero, the data-in buffer and/or data-out buffer associated with those commands shall consist of logical blocks with only user data; or
b) a non-zero value, the data-in buffer and/or data-out buffer shall consist of logical blocks with both user data and protection information.

### 4.17.2.5 Type 3 protection

Type 3 protection:

a) defines the content of the LOGICAL BLOCK GUARD field within the logical blocks of the data-in buffer and/or data-out buffer;
b) does not define the content of the LOGICAL BLOCK APPLICATION TAG field; and
c) does not define the content of the LOGICAL BLOCK REFERENCE TAG field.

If type 3 protection is enabled, then the following media commands are invalid and shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

a) READ (32);
b) VERIFY (32);
c) WRITE (32);
d) WRITE AND VERIFY (32); and
e) WRITE SAME (32).

For valid media access commands in which the RDPROTECT field, WRPROTECT field, ~~or~~ VRPROTECT field is set to:

a) zero, the data-in buffer and/or data-out buffer associated with those commands shall consist of logical blocks with only user data; or

b) a non-zero value, the data-in buffer and/or data-out buffer shall consist of logical blocks with both user data and protection information.

Add the following row to Table 12

**Table 12 — Commands for direct-access block devices**

| Command name | Operation code [a] | Type [b] | Protection information | Reference |
|---|---|---|---|---|
| ORWRITE (16) | xxh | O | Yes | 5.xx |

The following operation codes are vendor-specific:
02h, 05h, 06h, 09h, 0Ch, 0Dh, 0Eh, 0Fh,
10h, 11h, 13h, 14h, 19h,
20h, 21h, 22h, 23h, 24h, 26h, 27h, 29h, 2Ch, 2Dh, and
C0h through FFh.

All operation codes for direct-access block devices not specified in this table are reserved for future standardization.

[a] Some commands are defined by a combination of operation code and service action. The operation code value is shown preceding the slash and the service action value is shown after the slash.

[b] M = command implementation is mandatory. O = command implementation is optional. X = Command implementation requirements detailed in the reference.

[c] Application clients should migrate from READ (6) to READ (10) (see 5.6) and from WRITE (6) to WRITE (10) (see 5.25).

[d] READ CAPACITY (16) is mandatory if protection information is supported and optional otherwise.

[e] If the SCCS bit is set to one in the standard INQUIRY data (see SPC-4), these commands shall be supported as required by SCC-2. If the SCCS bit is set to zero, these commands shall not be supported.

[f] This command shall be supported if the ENCSERV bit is set to one in the standard INQUIRY data (see SPC-4) and may be supported otherwise.

Add the following command definition:

### 5.XX ORWRITE (16) command

The ORWRITE (16) command (see table yy1) requests that the device server perform the following as an uninterrupted series of actions:

1) read the specified logical block(s);
2) transfer logical blocks from the data-out buffer;
3) perform an OR operation with the logical blocks transferred from the data-out buffer and the logical blocks read, storing the resulting OR data in a buffer; and
4) write the logical blocks from the buffer.

Each logical block includes user data and may include protection information, based on the wrprotect field and the medium format.

**Table yy1 — ORWRITE (16) command**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (xxh) | | | | | | | |
| 1 | ORPROTECT | | | DPO | FUA | Reserved | FUA_NV | Reserved |
| 2 | (MSB) | LOGICAL BLOCK ADDRESS | | | | | | |
| 9 | | | | | | | | (LSB) |
| 10 | (MSB) | TRANSFER LENGTH | | | | | | |
| 13 | | | | | | | | (LSB) |
| 14 | Reserved | | | GROUP NUMBER | | | | |
| 15 | CONTROL | | | | | | | |

See the WRITE (10) command (see 5.26) for the definitions of the FUA bit and the FUA_NV bit. See the READ (10) command (see 5.7) for the definition of the DPO bit. See the PRE-FETCH (10) command (see 5.3) for the definition of the logical block address field. See the PRE-FETCH (10) command (see 5.3) and 4.17 for the definition of the group number field.

The transfer length field specifies the number of contiguous logical blocks of data that are read, transferred from the data-out buffer, and ORed into a buffer, starting with the logical block specified by the logical block address field. If the logical block address plus the transfer length exceeds the capacity of the medium, 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 transfer length field is constrained by the maximum transfer length field in the Block Limits VPD page (see 6.4.2).

The device server shall:

a) check protection information read from the medium based on the ORPROTECT field as described in table 2; and
b) check protection information transferred from the data-out buffer based on the ORPROTECT field as described in table yy2.

If the check of protection information read form the medium and check protection information transferred from the data-out buffer is successful, then the device server shall set the protection information (see 4.17) as it writes each logical block to the medium as follows:

a) the LOGICAL BLOCK GUARD field set to a CRC properly generated (see 4.17.4) by the device server;
b) the LOGICAL BLOCK REFERENCE TAG field set to the same LOGICAL BLOCK REFERENCE TAG field received from the data-out buffer; and
c) the LOGICAL BLOCK APPLICATION TAG field set to the same LOGICAL BLOCK APPLICATION TAG field received from the data-out buffer.

The order of the user data and protection information checks and comparisons is vendor-specific.

The device server shall check the protection information read from the medium based on the ORPROTECT field as described in table yy2.

**Table yy2 — ORPROTECT field checking protection information read from the medium** (part 1 of 2)

| Code | Logical unit formatted with protection information | Field in protection information [m] | Extended INQUIRY Data VPD page bit value [l] | If check fails [j] [k], additional sense code |
|---|---|---|---|---|
| 000b | Yes | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | GRD_CHK = 0 | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [i] [m] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [q] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | No protection information on the medium available to check | | |
| 001b<br>010b<br>011b<br>100b<br>101b [h] | Yes | LOGICAL BLOCK GUARD | No check performed | |
| | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No | Error condition [g] | | |
| 110b - 111b | Reserved | | | |

**Table yy2 — ORPROTECT field checking protection information read from the medium** (part 2 of 2)

| Code | Logical unit formatted with protection information | Field in protection information [m] | Extended INQUIRY Data VPD page bit value [l] | If check fails [j] [k], additional sense code |
|---|---|---|---|---|

[g] An ORWRITE command to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[h] If the logical unit does not support protection information the requested command should be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[i] The device server shall check the logical block application tag if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. This knowledge may be obtained by a method not defined by this standard.

[j] If an error is reported, the sense key shall be set to ABORTED COMMAND.

[k] If multiple errors occur, the selection of which error to report is not defined by this standard.

[l] See the Extended INQUIRY Data VPD page (see SPC-4) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bit.

[m] If the device server detects a:

[n] LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.17.2.3) is enabled; or

[o] LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF FFFFh, and type 3 protection (see 4.17.2.5) is enabled,

[p] then the device server shall not check any protection information in the associated logical block.

[q] If type 1 protection is enabled, the device server checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If type 3 protection is enabled, the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

The device server shall check the protection information transferred from the data-out buffer based on the ORPROTECT field as described in table yy3.

**Table yy3 — ORPROTECT field checking protection information from the data-out buffer** (part 1 of 3)

| Code | Logical unit formatted with protection information | Field in protection information | Device server check | If check fails [u] [v], additional sense code |
|---|---|---|---|---|
| 000b | Yes | No protection information received from application client to check | | |
| | No | No protection information received from application client to check | | |
| 001b [s] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | May [t] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | Shall [w] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [r] | | |

**Table yy3 —** ORPROTECT **field checking protection information from the data-out buffer** (part 2 of 3)

| Code | Logical unit formatted with protection information | Field in protection information | Device server check | If check fails [u] [v], additional sense code |
|---|---|---|---|---|
| 010b [s] | Yes | LOGICAL BLOCK GUARD | Shall not | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | May [t] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | May [w] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [r] | | |
| 011b [s] | Yes | LOGICAL BLOCK GUARD | Shall not | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | Shall not | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | Shall not | No check performed |
| | No | Error condition [r] | | |
| 100b [s] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | Shall not | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | Shall not | No check performed |
| | No | Error condition [r] | | |

**Table yy3 — ORPROTECT field checking protection information from the data-out buffer** (part 3 of 3)

| Code | Logical unit formatted with protection information | Field in protection information | Device server check | If check fails [u] [v], additional sense code |
|---|---|---|---|---|
| 101b [s] | Yes | logical block guard | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | logical block application tag | May [t] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | logical block reference tag | May [w] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [r] | | |
| 110b - 111b | Reserved | | | |

[r] An ORWRITE command to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[s] If the logical unit does not support protection information the requested command should be terminated with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[t] The device server may check the logical block application tag if the ATO bit is set to one in the Control mode page (see SPC-4) and if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. This knowledge is obtained by a method not defined by this standard.

[u] If an error is reported, the sense key shall be set to ABORTED COMMAND.

[v] If multiple errors occur, the selection of which error to report is not defined by this standard.

[w] If type 1 protection is enabled, the device server shall check the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. If type 3 protection is enabled, the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.