**T10/08-415 revision 4**

Date: 1/14/09

To: T10 Committee (SCSI)

From: George Penokie (LSI)

Subject: SBC-3/SPC-4 : Adding a Protection Information Interval

# 1 Overview

As logical block sizes get larger there is a concern that the CRC defined in the LOGICAL BLOCK GUARD field is not robust enough to adequately find errors in these larger logical blocks. This proposal adds the option to place protection information at regular intervals within a single logical block for types 1, 2, and 3 protection.

This proposal also leaves room for, but does not define, the option of defining new protection types that would span multiple logical blocks.

# 2 SPC-4 changes

Add a P_I_I_SUP bit into a bit position (editors choice) of the Extended INQUIRY Data VPD page.

The protection information interval supported (P_I_I_SUP) bit set to one indicates that the logical unit supports protection information intervals (SBC-3). A P_I_I_SUP bit set to zero indicates that the logical unit does not support protection information intervals.

# 3 SBC-3 changes

**3.0.1 protection information:** Fields appended to each logical block or added at specified intervals within a logical block that contain a cyclic redundancy check (CRC), an application tag, and a reference tag. See 4.17.

**3.0.2 protection information interval:** The length of user data that occurs within a logical block before each protection information.

## 3.1 Protection information model

### 3.1.1 Protection information overview

The protection information model provides for protection of user data while it is being transferred between a sender and a receiver. Protection information is generated at the application layer and may be checked by any object associated with the I_T_L nexus (see SAM-4). Once received, protection information is retained (e.g., written to medium, stored in non-volatile memory, or recalculated on read back) by the device server until overwritten. Power loss, hard reset, logical unit reset, and I_T nexus loss shall have no effect on the retention of protection information.

Support for protection information shall be indicated in the PROTECT bit in the standard INQUIRY data (see SPC-4).

If the logical unit is formatted with protection information and the EMDP bit is set to one in the Disconnect-Reconnect mode page (see SPC-4), then checking of the logical block reference tag within a service delivery subsystem without accounting for modified data pointers and data alignments may cause false errors when logical blocks are transmitted out of order.

Protection information is also referred to as the data integrity field (DIF).

### 3.1.2 Protection types

#### 3.1.2.1 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 5.13).

An application client may format the logical unit to a specific type of protection using the FMTPINFO field and the PROTECTION FIELD USAGE field in the FORMAT UNIT command (see 3.1.8.1).

An application client may format the logical unit to place protection information at intervals other than on logical block boundaries using the PROTECTION INTERVAL EXPONENT field in the FORMAT UNIT command (see 3.1.8.2).

The medium 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 “medium access commands” is defined as the following commands:

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

The device server may allow the READ (6) command (see 5.7) and the WRITE (6) command (see 5.26) regardless of the type of protection to which the logical unit has been formatted.

#### 3.1.2.2 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, VRPROTECT field, or ORPROTECT field is set to a non-zero value, then media commands are invalid and may be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If type 0 protection is enabled and the RDPROTECT field, WRPROTECT field, VRPROTECT field, or ORPROTECT field is set to a zero value, then the following media commands are invalid and shall be terminated by the device server 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).

### 3.1.2.3 Type 1 protection

Type 1 protection:

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

If type 1 protection is enabled, then the following media commands are invalid and shall be terminated by the device server 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 medium access commands in which the RDPROTECT field, WRPROTECT field, VRPROTECT field, 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.

### 3.1.2.4 Type 2 protection

Type 2 protection:

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

If type 2 protection is enabled and the RDPROTECT field, WRPROTECT field, VRPROTECT field, or ORPROTECT field is set to a non-zero value, then the following media commands are invalid and shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE:

a) ORWRITE;
b) READ (10);

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

For valid medium access commands in which the RDPROTECT field, WRPROTECT field, VRPROTECT field, 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.

#### 3.1.2.5 Type 3 protection

Type 3 protection:

a) defines the content of ~~the~~each 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~~any LOGICAL BLOCK APPLICATION TAG field; and
c) does not define the content of ~~the~~any LOGICAL BLOCK REFERENCE TAG field.

If type 3 protection is enabled, then the following media commands are invalid and shall be terminated by the device server 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 medium access commands in which the RDPROTECT field, WRPROTECT field, VRPROTECT field, 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.

### 3.1.3 Protection information format

Table 1 defines the placement of protection information in a logical block with a single protection information interval (i.e., PROTECTION INTERVAL EXPONENT field is set to zero in the FORMAT UNIT command (see 3.1.8.2)).

**Table 1 — User data and protection information format with a single protection information interval**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | USER DATA | | | | | | | |
| n - 1 | | | | | | | | |
| n | (MSB) | LOGICAL BLOCK GUARD | | | | | | |
| n + 1 | | | | | | | | (LSB) |
| n + 2 | (MSB) | LOGICAL BLOCK APPLICATION TAG | | | | | | |
| n + 3 | | | | | | | | (LSB) |
| n + 4 | (MSB) | LOGICAL BLOCK REFERENCE TAG | | | | | | |
| n + 7 | | | | | | | | (LSB) |

Table 2 shows an example of the placement of protection information in a logical block with more than one protection information interval (i.e., PROTECTION INTERVAL EXPONENT field is set to a non-zero value in the FORMAT UNIT command (see 3.1.8.2)).

**Table 2 — Example of user data and protection information format with more than one protection information interval**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | USER DATA (first) | | | | | | | |
| n - 1 | | | | | | | | |
| n | (MSB) | LOGICAL BLOCK GUARD (first) | | | | | | |
| n + 1 | | | | | | | | (LSB) |
| n + 2 | (MSB) | LOGICAL BLOCK APPLICATION TAG (first) | | | | | | |
| n + 3 | | | | | | | | (LSB) |
| n + 4 | (MSB) | LOGICAL BLOCK REFERENCE TAG (first) | | | | | | |
| n + 7 | | | | | | | | (LSB) |
| n + 8 | USER DATA (second) | | | | | | | |
| m - 1 | | | | | | | | |
| m | (MSB) | LOGICAL BLOCK GUARD (second) | | | | | | |
| m + 1 | | | | | | | | (LSB) |
| m + 2 | (MSB) | LOGICAL BLOCK APPLICATION TAG (second) | | | | | | |
| m + 3 | | | | | | | | (LSB) |
| m + 4 | (MSB) | LOGICAL BLOCK REFERENCE TAG (second) | | | | | | |
| m + 7 | | | | | | | | (LSB) |
| | . . . | | | | | | | |
| | USER DATA (last) | | | | | | | |
| z - 1 | | | | | | | | |
| z | (MSB) | LOGICAL BLOCK GUARD (last) | | | | | | |
| z + 1 | | | | | | | | (LSB) |
| z + 2 | (MSB) | LOGICAL BLOCK APPLICATION TAG (last) | | | | | | |
| z + 3 | | | | | | | | (LSB) |
| z + 4 | (MSB) | LOGICAL BLOCK REFERENCE TAG (last) | | | | | | |
| z + 7 | | | | | | | | (LSB) |

~~The~~Each USER DATA field shall contain user data. ~~The contents of the USER DATA field shall be used to generate and check the CRC contained in the LOGICAL BLOCK GUARD field.~~

~~The~~Each LOGICAL BLOCK GUARD field contains ~~the~~a CRC (see 3.1.4). Only ~~of~~ the contents of the USER DATA field immediately preceding THE LOGICAL BLOCK GUARD field. (i.e., the user data between the preceding logical block reference tag, if any, and the current logical block guard) shall be used to generate and check the CRC contained in the LOGICAL BLOCK GUARD field.

~~The~~Each LOGICAL BLOCK APPLICATION TAG field is set by the application client. If the device server detects a:

a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 3.1.2.3) is enabled;
b) ~~or~~ LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 2 protection (see 3.1.2.4) is enabled; or
c) LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF_FFFFh, and type 3 protection (see 3.1.2.5) is enabled,

then the device server disables checking of all protection information for ~~the logical block~~the associated protection information interval when reading from the medium. Otherwise, the contents of ~~the~~ logical block application tags are not defined by this standard.

The LOGICAL BLOCK APPLICATION TAG field may be modified by a device server if the ATO bit is set to zero in the Control mode page (see SPC-4). If the ATO bit is set to one in the Control mode page, then the device server shall not modify the LOGICAL BLOCK APPLICATION TAG field.

The contents of ~~the~~a LOGICAL BLOCK APPLICATION TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

The ~~-~~first LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer shall contain the value specified in table 3.

**Table 3 — Contents of the first LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer**

| Protection Type | Content of the ~~-~~first LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer |
|---|---|
| Type 1 protection [a] (see 3.1.2.3) | The least significant four bytes of the LBA contained in the LOGICAL BLOCK ADDRESS field of the command. |
| Type 2 protection (see 3.1.2.4) | The value in the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field of the command. |
| Type 3 protection (see 3.1.2.5) | Not defined in this standard. However, this field may be modified by the device server if the ATO bit is set to zero in the Control mode page (see SPC-4). If the ATO bit is set to one in the Control mode page, then the device server shall not modify this field. |

[a] The length of the protection information interval is equal to the logical block length (see 3.1.8.1).

~~The~~Subsequent LOGICAL BLOCK REFERENCE TAG fields ~~subsequent logical blocks~~ in the data-in buffer and/or data-out buffer shall be set as specified in table 4.

**Table 4 — Setting the ~~-~~subsequent LOGICAL BLOCK REFERENCE TAG fields ~~of the subsequent logical blocks~~ in the data-in buffer and/or data-out buffer**

| Protection Type | The content of ~~the~~subsequent LOGICAL BLOCK REFERENCE TAG fields ~~of each subsequent logical block~~ in the data-in buffer and/or data-out buffer |
|---|---|
| Type 1 protection (see 3.1.2.3) and Type 2 protection (see 3.1.2.4) | The previous logical block reference tag ~~of the previous logical block~~ plus one. |
| Type 3 protection (see 3.1.2.5) | Not defined in this standard. However, this field may be modified by the device server if the ATO bit is set to zero in the Control mode page (see SPC-4). If the ATO bit is set to one in the Control mode page, then the device server shall not modify this field. |

The contents of ~~the~~a LOGICAL BLOCK REFERENCE TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

### 3.1.4 Logical block guard

#### 3.1.4.1 Logical block guard overview

~~The~~A LOGICAL BLOCK GUARD field shall contain a CRC that is generated from the contents of only the USER DATA field immediately preceding the LOGICAL BLOCK GUARD field.

Table 5 defines the CRC polynomials used to generate the logical block guard from the contents of the USER DATA field.

**Table 5 — CRC polynomials**

| Function | Definition |
|---|---|
| F(x) | A polynomial representing the transmitted USER DATA field, which is covered by the CRC. For the purposes of the CRC, the coefficient of the highest order term shall be byte zero bit seven of the USER DATA field and the coefficient of the lowest order term shall be bit zero of the last byte of the USER DATA field. |
| F'(x) | A polynomial representing the received USER DATA field. |
| G(x) | The generator polynomial:<br>$G(x) = x^{16} + x^{15} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$<br>(i.e., G(x) = 18BB7h) |
| R(x) | The remainder polynomial calculated during CRC generation by the transmitter, representing the transmitted LOGICAL BLOCK GUARD field. |
| R'(x) | A polynomial representing the received LOGICAL BLOCK GUARD field. |
| RB(x) | The remainder polynomial calculated during CRC checking by the receiver.<br>RB(x) = 0 indicates no error was detected. |
| RC(x) | The remainder polynomial calculated during CRC checking by the receiver.<br>RC(x) = 0 indicates no error was detected. |
| QA(x) | The quotient polynomial calculated during CRC generation by the transmitter. The value of QA(x) is not used. |
| QB(x) | The quotient polynomial calculated during CRC checking by the receiver. The value of QB(x) is not used. |
| QC(x) | The quotient polynomial calculated during CRC checking by the receiver. The value of QC(x) is not used. |
| M(x) | A polynomial representing the transmitted USER DATA field followed by the transmitted LOGICAL BLOCK GUARD field. |
| M'(x) | A polynomial representing the received USER DATA field followed by the received LOGICAL BLOCK GUARD field. |

### 3.1.4.2 CRC generation

The equations that are used to generate the CRC from F(x) are as follows. All arithmetic is modulo 2.

The transmitter shall calculate the CRC by appending 16 zeros to F(x) and dividing by G(x) to obtain the remainder R(x):

$$\frac{(x^{16} \times F(x))}{G(x)} = QA(x) + \frac{R(x)}{G(x)}$$

R(x) is the CRC value, and is transmitted in the LOGICAL BLOCK GUARD field.

M(x) is the polynomial representing the USER DATA field followed by the LOGICAL BLOCK GUARD field (i.e., F(x) followed by R(x)):

$M(x) = (x^{16} \times F(x)) + R(x)$

#### 3.1.4.3 CRC checking

M’(x) (i.e., the polynomial representing the received USER DATA field followed by the received LOGICAL BLOCK GUARD field) may differ from M(x) (i.e., the polynomial representing the transmitted USER DATA field followed by the transmitted LOGICAL BLOCK GUARD field) if there are transmission errors.

The receiver may check M’(x) validity by appending 16 zeros to F’(x) and dividing by G(x) and comparing the calculated remainder RB(x) to the received CRC value R’(x):

$$\frac{(x^{16} \times F'(x))}{G(x)} = QB(x) + \frac{RB(x)}{G(x)}$$

In the absence of errors in F’(x) and R’(x), the remainder RB(x) is equal to R’(x).

The receiver may check M’(x) validity by dividing M’(x) by G(x) and comparing the calculated remainder RC(x) to zero:

$$\frac{M'(x)}{G(x)} = QC(x) + \frac{RC(x)}{G(x)}$$

In the absence of errors in F’(x) and R’(x), the remainder RC(x) is equal to zero.

Both methods of checking M’(x) validity are mathematically equivalent.

#### 3.1.4.4 CRC test cases

Several CRC test cases are shown in table 6.

**Table 6 — CRC test cases**

| Pattern | CRC |
|---|---|
| 32 bytes each set to 00h | 0000h |
| 32 bytes each set to FFh | A293h |
| 32 bytes of an incrementing pattern from 00h to 1Fh | 0224h |
| 2 bytes each set to FFh followed by 30 bytes set to 00h | 21B8h |
| 32 bytes of a decrementing pattern from FFh to E0h | A0B7h |

### 3.1.5 Application of protection information

Before an application client transmits or receives logical blocks with protection information it shall:

1) determine if a logical unit supports protection information using the INQUIRY command (see the PROTECT bit in the standard INQUIRY data in SPC-4);
2) if protection information is supported, then determine if the logical unit is formatted to accept protection information using the READ CAPACITY (16) command (see the PROT_EN bit in 5.13); and
3) if the logical unit supports protection information and is not formatted to accept protection information, then format the logical unit with protection information enabled.

If the logical unit supports protection information and is formatted to accept protection information, then the application client may use commands performing read operations that support protection information and should use commands performing write and verify operations that support protection information.

### 3.1.6 Protection information and commands

The enabling of protection information enables fields in some commands that instruct the device server on the handling of protection information. The detailed definitions of each command’s protection information fields are in the individual command descriptions.

The commands that are affected when protection information is enabled are listed in table 13 (see 5.1).

Commands that cause a device server to return the length in bytes of each logical block (e.g., the MODE SENSE commands and the READ CAPACITY commands) shall cause the device server to return the combined length of the USER DATA field(s) contained in the logical block, not including the length ~~of the~~any protection information (i.e., the LOGICAL BLOCK GUARD field(s), the LOGICAL BLOCK APPLICATION TAG field(s), and the LOGICAL BLOCK REFERENCE TAG field(s)) (e.g., if the user data plus the protection information is equal to 520 bytes and there is one protection information interval, then 512 is returned).

### 3.1.7 FORMAT UNIT command overview

...

### 3.1.8 FORMAT UNIT parameter list

#### 3.1.8.1 FORMAT UNIT parameter list overview

;;;

#### 3.1.8.2 Parameter list header

The parameter list headers (see table 7 and table 8) provide several optional format control parameters. Device servers that implement these headers provide the application client additional control over the use of the four defect sources, and the format operation. If the application client attempts to select any function not implemented by the device server, 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 INVALID FIELD IN PARAMETER LIST.

If the LONGLIST bit is set to zero in the FORMAT UNIT CDB, then the short parameter list header (see table 7) is used.

**Table 7 — Short parameter list header**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | | | | | PROTECTION FIELD USAGE | | |
| 1 | FOV | DPRY | DCRT | STPF | IP | Obsolete | IMMED | Vendor-specific |
| 2 | (MSB) | DEFECT LIST LENGTH | | | | | | |
| 3 | | | | | | | | (LSB) |

If the LONGLIST bit is set to one in the FORMAT UNIT CDB, then the long parameter list header (see table 8) is used.

**Table 8 — Long parameter list header**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | Reserved | | | | | PROTECTION FIELD USAGE | | |
| 1 | FOV | DPRY | DCRT | STPF | IP | Obsolete | IMMED | Vendor-specific |
| 2 | Reserved | | | | | | | |
| 3 | ~~Reserved~~P_I_INFORMATION | | | | PROTECTION INTERVAL EXPONENT | | | |
| 4 | (MSB) | | | | | | | |
| | DEFECT LIST LENGTH | | | | | | | |
| 7 | | | | | | | | (LSB) |

The PROTECTION FIELD USAGE field in combination with the FMTPINFO field (see table 9) specifies the requested protection type (see 4.17.2).

**Table 9 — FMTPINFO field and PROTECTION FIELD USAGE field** (part 1 of 2)

| Device server indication | | Application client specification | | Description |
|---|---|---|---|---|
| SPT [a] | PROTECT [b] | FMTPINFO | PROTECTION FIELD USAGE | |
| xxxb | 0 | 00b | 000b | The logical unit shall be formatted to type 0 protection [c] (see 4.17.2.2) resulting in the P_TYPE field [d] being set to 000b. |
| xxxb | 0 | 00b | >000b | Illegal [e] |
| xxxb | 0 | 01b | xxxb | Illegal [f] |
| xxxb | 0 | 1xb | xxxb | Illegal [f] |
| xxxb | 1 | 00b | 000b | The logical unit shall be formatted to type 0 protection [c] (see 4.17.2.2) resulting in the P_TYPE field [d] being set to 000b. |
| xxxb | 1 | 00b | >000b | Illegal [e] |

[a] See the Extended INQUIRY Data VPD page (see SPC-4) for the definition of the SPT field.
[b] See the standard INQUIRY data (see SPC-4) for the definition of the PROTECT bit.
[c] The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header (see SPC-4).
[d] See the READ CAPACITY command (see 3.6.1) for the definition of the P_TYPE field.
[e] 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 INVALID FIELD IN PARAMETER LIST.
[f] 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 INVALID FIELD IN CDB.
[g] The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header plus eight bytes for each protection information interval (e.g., if the logical block length is ~~512~~2 048 and the PROTECTION INTERVAL EXPONENT field set to two, then there are four 512 byte protection information intervals each followed by eight bytes of protection information resulting in a formatted logical block length of 2 080 bytes)~~, then the formatted logical block length is 520)~~. Following a successful format, the PROT_EN bit in the READ CAPACITY (16) parameter data (see 3.6.1) indicates whether protection information (see 4.17) is enabled.

**Table 9 — FMTPINFO field and PROTECTION FIELD USAGE field** (part 2 of 2)

| Device server indication | | Application client specification | | Description |
|---|---|---|---|---|
| SPT [a] | PROTECT [b] | FMTPINFO | PROTECTION FIELD USAGE | |
| xxxb | 1 | 01b | xxxb | Illegal [f] |
| 000b<br>001b<br>011b | 1 | 10b | 000b | The logical unit shall be formatted to type 1 protection [g] (see 4.17.2.3) resulting in the P_TYPE field [d] being set to 000b. |
| 000b<br>001b<br>011b | 1 | 10b | >000b | Illegal [e] |
| 000b | 1 | 11b | xxxb | Illegal [f] |
| 001b | 1 | 11b | 000b | The logical unit shall be formatted to type 2 protection [g] (see 4.17.2.4) resulting in the P_TYPE field [d] being set to 001b. |
| 001b | 1 | 11b | >000b | Illegal [e] |
| 011b | 1 | 11b | 000b | Illegal [e] |
| 011b | 1 | 1 | 001b | The logical unit shall be formatted to type 3 protection. [g] (see 4.17.2.5) resulting in the P_TYPE field [d] being set to 010b. |
| 011b | 1 | 11b | >001b | Illegal [e] |
| 010b | 1 | 1xb | xxxb | Reserved |
| 1xxb | 1 | 1xb | xxxb | Reserved |

[a] See the Extended INQUIRY Data VPD page (see SPC-4) for the definition of the SPT field.
[b] See the standard INQUIRY data (see SPC-4) for the definition of the PROTECT bit.
[c] The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header (see SPC-4).
[d] See the READ CAPACITY command (see 3.6.1) for the definition of the P_TYPE field.
[e] 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 INVALID FIELD IN PARAMETER LIST.
[f] 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 INVALID FIELD IN CDB.
[g] The device server shall format the medium to the logical block length specified in the mode parameter block descriptor of the mode parameter header plus eight bytes for each protection information interval (e.g., if the logical block length is ~~512~~2 048 and the PROTECTION INTERVAL EXPONENT field set to two, then there are four 512 byte protection information intervals each followed by eight bytes of protection information resulting in a formatted logical block length of 2 080 bytes)~~, then the formatted logical block length is 520~~). Following a successful format, the PROT_EN bit in the READ CAPACITY (16) parameter data (see 3.6.1) indicates whether protection information (see 4.17) is enabled.

....

For a type 1 protection formation request if the PROTECTION INTERVAL EXPONENT field is not set to zero, 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 INVALID FIELD IN PARAMETER LIST.

For a type 2 protection or type 3 protection format requests the protection interval exponent determines the length of user data to be sent before protection information is transferred (i.e., the protection information interval).

The protection information interval is calculated as follows:

protection information interval = logical block length / $2^{(\text{protection interval exponent})}$

where:

| | |
|---|---|
| logical block length | is the length in bytes of a logical block as specified in the mode parameter block descriptor of the mode parameter header (see 6.3.2.1xx) |
| protection interval exponent | is the contents of the PROTECTION INTERVAL EXPONENT field |

If the protection information interval calculates to a value that is not an even number (e.g., $520/2^3 = 65$) or not a whole number (e.g., $520/2^4 = 32.5$ and $520/2^{10} = 0.508$), 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 INVALID FIELD IN PARAMETER LIST.

The P_I_INFORMATION field shall be set to zero.

## 3.2 ORWRITE command

...

If the check of protection information read from 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~~each LOGICAL BLOCK GUARD field set to a CRC properly generated (see 4.17.4) by the device server;
b) ~~the~~each LOGICAL BLOCK REFERENCE TAG field set to the same LOGICAL BLOCK REFERENCE TAG field received from the data-out buffer; and
c) ~~the~~each 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 10.

**Table 10 —** ORPROTECT **field - checking protection information read from the medium** (part 1 of 4)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d] [e], 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 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | No protection information on the medium to check. Only user data is checked. | | |

**Table 10 —** ORPROTECT **field - checking protection information read from the medium** (part 2 of 4)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d] [e], additional sense code |
|---|---|---|---|---|
| 001b<br>101b [b] | 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 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | Error condition [a] | | |
| 010b [b] | Yes | LOGICAL BLOCK GUARD | No check performed | |
| | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | Error condition [a] | | |

**Table 10 —** ORPROTECT **field - checking protection information read from the medium** (part 3 of 4)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d] [e], additional sense code |
|---|---|---|---|---|
| 011b [b] | Yes | LOGICAL BLOCK GUARD | No check performed | |
| | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No | Error condition [a] | | |
| 100b [b] | Yes | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | GRD_CHK = 0 | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No | Error condition [a] | | |
| 110b to 111b | Reserved | | | |

**Table 10 —** ORPROTECT **field - checking protection information read from the medium** (part 4 of 4)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d] [e], additional sense code |
|---|---|---|---|---|
| [a] 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 by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.<br>[b] If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.<br>[c] 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.<br>[d] If the device server terminates the command with CHECK CONDITION status, then the sense key shall be set to ABORTED COMMAND.<br>[e] If multiple errors occur, the selection of which error to report is not defined by this standard.<br>[f] 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 bits.<br>[g] If the device server detects a:<br>a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.17.2.3) is enabled; or<br>b) 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,<br>then the device server shall not check any protection information in the associated ~~logical block~~protection information interval.<br>[h] If type 1 protection is enabled, then 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, then the device server checks ~~the~~each logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. 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 11.

**Table 11 —** 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 [d] [e], 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 [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | May [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | Shall (except for type 3) [f] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [a] | | |

**Table 11 —** 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 [d] [e], additional sense code |
|---|---|---|---|---|
| 010b [b] | Yes | LOGICAL BLOCK GUARD | Shall not | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | May [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | May [f] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [a] | | |
| 011b [b] | 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 [a] | | |
| 100b [b] | 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 [a] | | |
| 101b [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | May [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | May [f] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [a] | | |
| 110b to 111b | Reserved | | | |

**Table 11 —** 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 [d] [e], additional sense code |
|---|---|---|---|---|
| [a] An or write operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.<br>[b] If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.<br>[c] The device server may check ~~the~~each logical block application tag if the ATO bit is set to one in the Control mode page (see SPC-4) and if the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. This knowledge is obtained by a method not defined by this standard.<br>[d] If the device server terminates the command with CHECK CONDITION status, the sense key shall be set to ABORTED COMMAND.<br>[e] If multiple errors occur, the selection of which error to report is not defined by this standard.<br>[f] If type 1 protection is enabled, then 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 ATO bit is set to one in the Control mode page (see SPC-4), and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server may check ~~the~~each logical block reference tag. The method for acquiring this knowledge is not defined by this standard. | | | | |

## 3.3 READ (6) command

...

The device server shall check the protection information read from the medium before returning status for the command as described in table 12.

**Table 12 — Protection information checking for READ (6)**

| Logical unit formatted with protection information | Shall device server transmit protection information? | Field in protection information | Extended INQUIRY Data VPD page bit value [d] | If check fails [b] [c], additional sense code |
|---|---|---|---|---|
| Yes | No | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | GRD_CHK = 0 | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [a] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [f] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| No | | No protection information available to check | | |

[a] The device server checks ~~the~~each logical block application tag only if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. The method for acquiring this knowledge is not defined by this standard.

[b] If the device server terminates the command with CHECK CONDITION status, then the sense key shall be set to ABORTED COMMAND.

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

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

[e] If the device server detects a:

a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.17.2.3) or type 2 protection (see 4.17.2.4) is enabled; or

b) 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,

then the device server shall not check any protection information in the associated ~~logical block~~protection information interval.

[f] If type 1 protection is enabled, then 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 2 protection or type 3 protection is enabled, then the device server checks ~~the~~each logical block reference tag only if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. The method for acquiring this knowledge is not defined by this standard.

## 3.4 READ (10) command

...

The device server shall check the protection information read from the medium before returning status for the command based on the RDPROTECT field as described in table 13.

**Table 13 —** RDPROTECT **field** (part 1 of 3)

| Code | Logical unit formatted with protection information | Shall device server transmit protection information? | Field in protection information [h] | Extended INQUIRY Data VPD page bit value [g] | If check fails [d] [f], additional sense code |
|---|---|---|---|---|---|
| 000b | Yes | No | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | | GRD_CHK = 0 | No check performed |
| | | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | | APP_CHK = 0 | No check performed |
| | | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [i] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | | REF_CHK = 0 | No check performed |
| | No | | No protection information available to check | | |
| 001b<br>101b [b] | Yes | Yes [e] | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | | GRD_CHK = 0 | No check performed |
| | | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | | APP_CHK = 0 | No check performed |
| | | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [i] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | | REF_CHK = 0 | No check performed |
| | No [a] | No protection information available to transmit to the data-in buffer or for checking | | | |

**Table 13 —** RDPROTECT **field** (part 2 of 3)

| Code | Logical unit formatted with protection information | Shall device server transmit protection information? | Field in protection information [h] | Extended INQUIRY Data VPD page bit value [g] | If check fails [d] [f], additional sense code |
|---|---|---|---|---|---|
| 010b [b] | Yes | Yes [e] | LOGICAL BLOCK GUARD | No check performed | |
| | | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | | APP_CHK = 0 | No check performed |
| | | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [i] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | | REF_CHK = 0 | No check performed |
| | No [a] | No protection information available to transmit to the data-in buffer or for checking | | | |
| 011b [b] | Yes | Yes [e] | LOGICAL BLOCK GUARD | No check performed | |
| | | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No [a] | No protection information available to transmit to the data-in buffer or for checking | | | |
| 100b [b] | Yes | Yes [e] | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | | GRD_CHK = 0 | No check performed |
| | | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No [a] | No protection information available to transmit to the data-in buffer or for checking | | | |
| 110b to 111b | Reserved | | | | |

**Table 13 —** RDPROTECT **field** (part 3 of 3)

| Code | Logical unit formatted with protection information | Shall device server transmit protection information? | Field in protection information [h] | Extended INQUIRY Data VPD page bit value [g] | If check fails [d] [f], additional sense code |
|---|---|---|---|---|---|
| [a] A read operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.<br>[b] If the logical unit does not support protection information, then the device server should terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.<br>[c] The device server shall check ~~the~~each logical block application tag if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the READ (32) command (see 5.11) is used, and the ATO bit is set to one in the Control mode page (see SPC-4), then this knowledge is acquired from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge may be acquired by a method not defined by this standard.<br>[d] If the device server terminates the command with CHECK CONDITION status, then the sense key shall be set to ABORTED COMMAND.<br>[e] Transmit protection information to the data-in buffer.<br>[f] If multiple errors occur, the selection of which error to report is not defined by this standard.<br>[g] 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.<br>[h] If the device server detects a:<br>a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.17.2.3) or type 2 protection (see 4.17.2.4) is enabled; or<br>b) 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,<br>then the device server shall not check any protection information in the associated ~~logical block~~protection information interval.<br>[i] If type 1 protection is enabled, then 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 2 protection or type 3 protection is enabled, then the device server checks ~~the~~each logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a READ (32) command (see 5.11). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard. | | | | | |

## 3.5 READ (32) command

...

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 39 in 5.8), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in ~~the protection information~~ every instance of protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK field bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in ~~theprotection information~~every instance of protection information.

## 3.6 READ CAPACITY (16) command

### 3.6.1 READ CAPACITY (16) command overview

...

### 3.6.2 READ CAPACITY (16) parameter data

...

The READ CAPACITY (16) parameter data is defined in table 14. Any time the READ CAPACITY (16) parameter data changes, the device server should establish a unit attention condition as described in 4.7.

**Table 14 — READ CAPACITY (16) parameter data**

| Byte \ Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | | | | | | | |
| 7 | RETURNED LOGICAL BLOCK ADDRESS | | | | | | | (LSB) |
| 8 | (MSB) | | | | | | | |
| 11 | LOGICAL BLOCK LENGTH IN BYTES | | | | | | | (LSB) |
| 12 | Reserved | | | | P_TYPE | | | PROT_EN |
| 13 | ~~Reserved~~P_I_EXPONENT | | | | LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT | | | |
| 14 | Reserved | | (MSB) | | | | | |
| 15 | LOWEST ALIGNED LOGICAL BLOCK ADDRESS | | | | | | | (LSB) |
| 16 | | | | | | | | |
| 31 | Reserved | | | | | | | |

The RETURNED LOGICAL BLOCK ADDRESS field and LOGICAL BLOCK LENGTH IN BYTES field of the READ CAPACITY (16) parameter data are the same as the in the READ CAPACITY (10) parameter data (see 5.12). The maximum value that shall be returned in the RETURNED LOGICAL BLOCK ADDRESS field is FFFF_FFFF_FFFF_FFFEh.

The protection type (P_TYPE) field and the protection enable (PROT_EN) bit (see table 15) indicate the logical unit's current type of protection.

**Table 15 — P_TYPE field and PROT_EN bit**

| PROT_EN | P_TYPE | Description |
|---|---|---|
| 0 | xxxb | The logical unit is formatted to type 0 protection (see 4.17.2.2). |
| 1 | 000b | The logical unit is formatted to type 1 protection (see 4.17.2.3). |
| 1 | 001b | The logical unit is formatted to type 2 protection (see 4.17.2.4). |
| 1 | 010b | The logical unit is formatted to type 3 protection (see 4.17.2.5). |
| 1 | 011b to 111b | Reserved |

The LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field is defined in table 16.

**Table 16 — LOGICAL BLOCKS PER PHYSICAL BLOCK EXPONENT field**

| Code | Description |
|---|---|
| 0 | One or more physical blocks per logical block [a] |
| n > 0 | $2^n$ logical blocks per physical block |
| [a] The number of physical blocks per logical block is not reported. | |

The P_I_EXPONENT field may be used to determine the number of protection information intervals placed within each logical block (see 3.1.8).

The number of protection information intervals is calculated as follows:

$$\text{number of protection information intervals} = 2^{(\text{p\_i exponent})}$$

where:

p_i exponent is the contents of the P_I EXPONENT field

The LOWEST ALIGNED LOGICAL BLOCK ADDRESS field indicates the LBA of the first logical block that is located at the beginning of a physical block (see 4.5).

NOTE 1 - The highest LBA that the lowest aligned logical block address field supports is 3FFFh (i.e., 16 383).

## 3.7 VERIFY (10) command

...

If the BYTCHK bit is set to zero, then the device server shall check the protection information read from the medium based on the VRPROTECT field as described in table 17.

**Table 17 — VRPROTECT field with BYTCHK set to zero - checking protection information read from the medium** (part 1 of 3)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d] [e], 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 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | No protection information on the medium to check. Only user data is checked. | | |

**Table 17 — VRPROTECT field with BYTCHK set to zero - checking protection information read from the medium** (part 2 of 3)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d] [e], additional sense code |
|---|---|---|---|---|
| 001b<br>101b [b] | 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 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | Error condition [a] | | |
| 010b [b] | Yes | LOGICAL BLOCK GUARD | No check performed | |
| | | LOGICAL BLOCK APPLICATION TAG | APP_CHK = 1 [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [h] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | Error condition [a] | | |
| 011b [b] | Yes | LOGICAL BLOCK GUARD | No check performed | |
| | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No | Error condition [a] | | |
| 100b [b] | Yes | LOGICAL BLOCK GUARD | GRD_CHK = 1 | LOGICAL BLOCK GUARD CHECK FAILED |
| | | | GRD_CHK = 0 | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No | Error condition [a] | | |

**Table 17 — VRPROTECT field with BYTCHK set to zero - checking protection information read from the medium** (part 3 of 3)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d] [e], additional sense code |
|---|---|---|---|---|
| 110b to 111b | Reserved | | | |

[a] A verify operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[c] The device server shall check ~~the~~each logical block application tag if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the VERIFY (32) command (see 5.25) is used, and the ATO bit is set to one in the Control mode page (see SPC-4), then this knowledge is acquired from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge may be obtained by a method not defined by this standard.

[d] If the device server terminates the command with CHECK CONDITION status, then the sense key shall be set to ABORTED COMMAND.

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

[f] 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 bits.

[g] If the device server detects a:

a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.17.2.3) or type 2 protection (see 4.17.2.4) is enabled; or

b) 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,

then the device server shall not check any protection information in the associated ~~logical block~~protection information interval.

[h] If type 1 protection is enabled, then 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 2 protection or type 3 protection is enabled, then the device server checks ~~the~~each logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 5.25). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

If the BYTCHK bit is set to one, then the device server shall check the protection information read from the medium based on the VRPROTECT field as described in table 18.

**Table 18 —** VRPROTECT **field with** BYTCHK **set to one - checking protection information read from the medium** (part 1 of 2)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d] [e], 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 [c] [g] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | | APP_CHK = 0 | No check performed |
| | | LOGICAL BLOCK REFERENCE TAG | REF_CHK = 1 [h] | 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 [b] | Yes | LOGICAL BLOCK GUARD | No check performed | |
| | | LOGICAL BLOCK APPLICATION TAG | No check performed | |
| | | LOGICAL BLOCK REFERENCE TAG | No check performed | |
| | No | Error condition [a] | | |
| 110b to 111b | Reserved | | | |

**Table 18 —** VRPROTECT **field with** BYTCHK **set to one - checking protection information read from the medium** (part 2 of 2)

| Code | Logical unit formatted with protection information | Field in protection information [g] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d] [e], additional sense code |
|---|---|---|---|---|

[a] A verify operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[c] The device server shall check ~~the~~each logical block application tag if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the VERIFY (32) command (see 5.25) is used, and the ATO bit is set to one in the Control mode page (see SPC-4), then this knowledge is acquired from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge may be obtained by a method not defined by this standard.

[d] If the device server terminates the command with CHECK CONDITION status, then the sense key shall be set to ABORTED COMMAND.

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

[f] 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.

[g] If the device server detects a:

a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and type 1 protection (see 4.17.2.3) or type 2 protection (see 4.17.2.4) is enabled; or

b) 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,

then the device server shall not check any protection information in the associated ~~logical block~~protection information interval.

[h] If type 1 protection is enabled, then 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 2 protection or type 3 protection is enabled, then the device server checks ~~the~~each logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 5.25). If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

If the BYTCHK bit is set to one, then the device server shall check the protection information transferred from the data-out buffer based on the VRPROTECT field as described in table 19.

**Table 19 —** VRPROTECT **field with** BYTCHK **set to one - checking protection information from the data-out buffer** (part 1 of 2)

| Code | Logical unit formatted with protection information | Field in protection information | Device server check | If check fails [d] [e], 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 [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | May [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | Shall (except for type 3) [f] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [a] | | |
| 010b [b] | Yes | LOGICAL BLOCK GUARD | Shall not | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | May [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | May [f] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [a] | | |
| 011b [b] | 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 [a] | | |
| 100b [b] | 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 [a] | | |

**Table 19 —** VRPROTECT **field with** BYTCHK **set to one - checking protection information from the data-out buffer** (part 2 of 2)

| Code | Logical unit formatted with protection information | Field in protection information | Device server check | If check fails [d] [e], additional sense code |
|---|---|---|---|---|
| 101b [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | May [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | May [f] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [a] | | |
| 110b to 111b | Reserved | | | |

[a] A verify operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[c] The device server may check ~~the~~each logical block application tag if the ATO bit is set to one in the Control mode page (see SPC-4) and if the device server has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the VERIFY (32) command (see 5.25) is used, then this knowledge is obtained from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge is obtained by a method not defined by this standard.

[d] If the device server terminates the command with CHECK CONDITION status, then the device server shall set the sense key to ABORTED COMMAND.

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

[f] If type 1 protection is enabled, then 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 2 protection is enabled, and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server checks ~~the~~each logical block reference tag. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 5.25). If type 3 protection is enabled, the ATO bit is set to one in the Control mode page (see SPC-4), and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server may check ~~the~~each logical block reference tag. If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

If the BYTCHK bit is set to one, then the device server shall perform a byte-by-byte comparison of protection information transferred from the data-out buffer with protection information read from the medium based on the VRPROTECT field as described in table 20.

**Table 20 —** VRPROTECT **field with** BYTCHK **set to one - byte-by-byte comparison requirements** (part 1 of 2)

| Code | Logical unit formatted with protection information | Field | Byte-by-byte Comparison | If compare fails [c] [d], additional sense code |
|---|---|---|---|---|
| 000b | Yes | No protection information received from application client to compare. Only user data is compared within each logical block. | | |
| | No | No protection information or the medium or received from application client to compare. Only user data is compared within each logical block. | | |
| 001b [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 1) [e] | Shall | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f] | Shall not | No compare performed |
| | | LOGICAL BLOCK REFERENCE TAG (not type 3) | Shall | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 0) | Shall | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 1) | Shall not | No compare performed |
| | No | Error condition [a] | | |
| 010b [b] | Yes | LOGICAL BLOCK GUARD | Shall not | No compare performed |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 1) [e] | Shall | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f] | Shall not | No compare performed |
| | | LOGICAL BLOCK REFERENCE TAG (not type 3) | Shall | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 0) | Shall | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 1) | Shall not | No compare performed |
| | No | Error condition [a] | | |

**Table 20 —** VRPROTECT **field with** BYTCHK **set to one - byte-by-byte comparison requirements** (part 2 of 2)

| Code | Logical unit formatted with protection information | Field | Byte-by-byte Comparison | If compare fails [c] [d], additional sense code |
|---|---|---|---|---|
| 011b<br>100b [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 1) [e] | Shall | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f] | Shall not | No compare performed |
| | | LOGICAL BLOCK REFERENCE TAG (not type 3) | Shall | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 0) | Shall | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG (type 3 and ATO = 1) | Shall not | No compare performed |
| | No | Error condition [a] | | |
| 101b [b] | Yes | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 1) [e] | Shall | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG (ATO = 0) [f] | Shall not | No compare performed |
| | | LOGICAL BLOCK REFERENCE TAG | Shall not | No compare performed |
| | No | Error condition [a] | | |
| 110b to 111b | Reserved | | | |

[a] A verify operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[c] If the device server terminates the command with CHECK CONDITION status, then the sense key shall be set to MISCOMPARE.

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

[e] If the ATO bit is set to one in the Control mode page (see SPC-4), then the logical block application tag shall not be modified by a device server.

[f] If the ATO bit is set to zero in the Control mode page (see SPC-4), then ~~the~~any logical block application tag may be modified by a device server.

## 3.8 VERIFY (32) command

...

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 67, table 68, table 69, and table 70 in 5.22), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in ~~the protection information~~every instance of protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in ~~the protection information~~every instance of protection information.

## 3.9 WRITE (6) command

...

If a WRITE (6) command is received after protection information is enabled, then the device server shall set the protection information (see 4.17) as follows as the device server writes each logical block to the medium:

a) ~~the~~each LOGICAL BLOCK GUARD field set to a properly generated CRC (see 4.17.4);
b) ~~the~~each LOGICAL BLOCK REFERENCE TAG field set to:
   A) the least significant four bytes of the LBA, if type 1 protection (see 4.17.2.3) is enabled;
   B) FFFF_FFFFh, if type 2 protection is enabled (see 4.17.2.4);
   C) FFFF_FFFFh, if the ATO bit is set to one in the Control mode page (see SPC-4), and type 3 protection (see 4.17.2.5) is enabled; or
   D) any value, if the ATO bit is set to zero in the Control mode page (see SPC-4), and type 3 protection (see 4.17.2.5) is enabled;

   and
c) ~~the~~each LOGICAL BLOCK APPLICATION TAG field set to:
   A) FFFFh, if the ATO bit is set to one in the Control mode page (see SPC-4); or
   B) any value, if the ATO bit is set to zero in the Control mode page (see SPC-4).

## 3.10 WRITE (10) command

...

The device server shall check the protection information transferred from the data-out buffer based on the WRPROTECT field as described in table 21.

**Table 21 — WRPROTECT field** (part 1 of 3)

| Code | Logical unit formatted with protection information | Field in protection information | Device server check | If check fails [d] [i], additional sense code |
|---|---|---|---|---|
| 000b | Yes [f] [g] [h] | No protection information received from application client to check | | |
| | No | No protection information received from application client to check | | |

**Table 21 —** WRPROTECT **field** (part 2 of 3)

| Code | Logical unit formatted with protection information | Field in protection information | Device server check | If check fails [d] [i], additional sense code |
|---|---|---|---|---|
| 001b [b] | Yes [e] | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | May [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | Shall (except for type 3) [j] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No [a] | No protection information available to check | | |
| 010b [b] | Yes [e] | LOGICAL BLOCK GUARD | Shall not | No check performed |
| | | LOGICAL BLOCK APPLICATION TAG | May [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | May [j] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No [a] | No protection information available to check | | |
| 011b [b] | Yes [e] | 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 [a] | No protection information available to check | | |
| 100b [b] | Yes [e] | 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 [a] | No protection information available to check | | |
| 101b [b] | Yes [e] | LOGICAL BLOCK GUARD | Shall | LOGICAL BLOCK GUARD CHECK FAILED |
| | | LOGICAL BLOCK APPLICATION TAG | May [c] | LOGICAL BLOCK APPLICATION TAG CHECK FAILED |
| | | LOGICAL BLOCK REFERENCE TAG | May [j] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No [a] | No protection information available to check | | |
| 110b to 111b | Reserved | | | |

**Table 21 —** WRPROTECT **field** (part 3 of 3)

| Code | Logical unit formatted with protection information | Field in protection information | Device server check | If check fails [d] [i], additional sense code |
|---|---|---|---|---|

[a] A write operation to a logical unit that supports protection information (see 4.17) and has not been formatted with protection information shall be terminated by the device server with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[b] If the logical unit does not support protection information, then the device server should terminate the requested command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

[c] The device server may check ~~the~~each 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. If the WRITE (32) command (see 5.30) is used, then this knowledge is obtained from the EXPECTED LOGICAL BLOCK APPLICATION TAG field and the LOGICAL BLOCK APPLICATION TAG MASK field in the CDB. Otherwise, this knowledge is obtained by a method not defined by this standard.

[d] If the device server terminates the command with CHECK CONDITION status, then the sense key shall be set to ABORTED COMMAND.

[e] Device server shall preserve the contents of protection information (e.g., write to medium, store in non-volatile memory).

[f] The device server shall write a properly generated CRC (see 4.17.4.2) into each LOGICAL BLOCK GUARD field.

[g] If ~~the P_TYPE field is set to 000h in the READ CAPACITY (16) parameter data (see 5.13)~~type 1 protection is enabled, then the device server shall write the least significant four bytes of each LBA into the LOGICAL BLOCK REFERENCE TAG field of each of the written logical blocks. If ~~the P_TYPE field is not set to 000b~~type 2 protection or type 3 protection is enabled, then the device server shall write a value of FFFF_FFFFh into ~~the~~each LOGICAL BLOCK REFERENCE TAG field of each of the written logical blocks.

[h] If the ATO bit is set to one in the Control mode page (see SPC-4), then the device server shall write FFFFh into each LOGICAL BLOCK APPLICATION TAG field. If the ATO bit is set to zero, then the device server may write any value into each LOGICAL BLOCK APPLICATION TAG field.

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

[j] If type 1 protection is enabled, then 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 2 protection is enabled, and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server checks ~~the~~each logical block reference tag. If type 2 protection is enabled, then this knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a WRITE (32) command (see 5.30). If type 3 protection is enabled, the ATO bit is set to one in the Control mode page (see SPC-4), and the device server has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field, then the device server may check ~~the~~each logical block reference tag. If type 3 protection is enabled, then the method for acquiring this knowledge is not defined by this standard.

## 3.11 WRITE (32) command

...

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 21 in 3.10), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in ~~the protection information~~every instance of protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in ~~the protection information~~every instance of protection information.

## 3.12 WRITE AND VERIFY (32) command

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 21 in 3.10), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in ~~the protection information~~every instance of protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in ~~the protection information~~every instance of protection information.

## 3.13 WRITE SAME (10) command

...

Table 22 describes the lbdata bit and the pbdata bit.

**Table 22 — LBDATA bit and PBDATA bit**

| LBDATA | PBDATA | Description |
|---|---|---|
| 0 | 0 | The device server shall write the single block of user data received from the data-out buffer to each logical block without modification.<br><br>If the medium is formatted with type 1 or type 2 protection information, then:<br>a) the value in ~~the~~each LOGICAL BLOCK REFERENCE TAG field received in the single block of data from the data-out buffer shall be placed into the ~~corre-~~sponding LOGICAL BLOCK REFERENCE TAG field of the first logical block written to the medium. Into each of the subsequent LOGICAL BLOCK REFERENCE TAG fields ~~logical blocks~~, the device server shall place ~~into the LOGICAL BLOCK REFERENCE TAG field~~ the value of the previous ~~logical block's~~ LOGICAL BLOCK REFERENCE TAG field plus one.<br>b) If the ATO bit is set to one in the Control mode page (see SPC-4), then ~~the~~each logical block application tag received in the single block of data shall be placed in the ~~corresponding~~ LOGICAL BLOCK APPLICATION TAG field of each logical block. If the ATO bit is set to zero, then the device server may write any value into the LOGICAL BLOCK APPLICATION TAG field(s) of each logical block; and<br>c) The value in ~~the~~each LOGICAL BLOCK GUARD field received in the single block of data from the data-out buffer shall be placed in the ~~corresponding~~ LOGICAL BLOCK GUARD field of each logical block.<br><br>If the medium is formatted with type 3 protection information:<br>a) If the ATO bit is set to one in the Control mode page (see SPC-4), then ~~the~~each logical block reference tag received in the single block of data shall be placed in the ~~corresponding~~ LOGICAL BLOCK REFERENCE TAG field of each logical block. If the ATO bit is set to zero, then the device server may write any value into the LOGICAL BLOCK REFERENCE TAG field(s) of each logical block;<br>b) If the ATO bit is set to one in the Control mode page (see SPC-4), then ~~the~~each logical block application tag received in the single block of data shall be placed in the ~~corresponding~~ LOGICAL BLOCK REFERENCE TAG field of each logical block. If the ATO bit is set to zero, then the device server may write any value into the LOGICAL BLOCK REFERENCE TAG field(s) of each logical block; and<br>c) The value in ~~the~~each LOGICAL BLOCK GUARD field received in the single block of data from the data-out buffer shall be placed in the ~~corresponding~~ LOGICAL BLOCK GUARD field of each logical block. |
| 0 | 1 [a] | The device server shall replace the first eight bytes of the block received from the data-out buffer to each physical sector with the physical address of the sector being written using the physical sector format (see 5.2.2.4.5). |
| 1 [a] | 0 | The device server shall replace the first four bytes of the block received from the data-out buffer with the least significant four bytes of the LBA of the block being written, ending with the least significant byte (e.g., if the LBA is 7766_5544_3322_1100h, 3322_1100h is written with 33h written first and 00h written last). |

[a] If the medium is formatted with protection information, then all the protection information shall be written to a default value of FFFF_FFFF_FFFF_FFFFh in each of the written logical blocks.

**Table 22 — LBDATA bit and PBDATA bit**

| LBDATA | PBDATA | Description |
|---|---|---|
| 1 | 1 | 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 INVALID FIELD IN CDB. |

[a] If the medium is formatted with protection information, then all the protection information shall be written to a default value of FFFF_FFFF_FFFF_FFFFh in each of the written logical blocks.

## 3.14 WRITE SAME (32) command

If the ATO bit is set to one in the Control mode page (see SPC-4) and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 21 in 3.10), then the LOGICAL BLOCK APPLICATION TAG MASK field contains a value that is a bit mask for enabling the checking of the LOGICAL BLOCK APPLICATION TAG field in ~~the protection information~~every instance of protection information for each logical block accessed by the command. A LOGICAL BLOCK APPLICATION TAG MASK bit set to one enables the checking of the corresponding bit of the EXPECTED LOGICAL BLOCK APPLICATION TAG field with the corresponding bit of the LOGICAL BLOCK APPLICATION TAG field in ~~the protection information~~every instance of protection information.