Date: August 31, 2005

To: T10 Committee (SCSI)

From: George Penokie (IBM/Tivoli)

Subject: SBC-3, SPC-4: Application ownership of protection information Reference Tag

## 1 Overview

Some of the current applications using proprietary end-to-end protection method would like to convert to using the protection defined in the SBC and SPC standards. However, the methods used require a larger application tag field than is currently defined. They also imbed information that performs a similar function to the reference tag field. As a result they would like to have the option to expand the application tag field to include the reference tag field.

The following proposal requests the RTO_EN bit be expanded to a 3 bit field to allow different usages of the reference tag field (e.g., setting the RTO_EN field to 010b would have the effect of preventing the device server from modifying the reference tag). A bit is also added to Extended INQUIRY Data VPD page to inform the application client if this option is supported.

With this proposal the meaning of a logical unit that is formatted with RTO_EN set to one changes from failing all non-32-byte command except legacy for legacy operations (i.e., those with RDPROTECT, WRPROTECT, and VRPROTECT set to zero) to allowing all commands to execute.

Because the 32-byte commands now have two different ways to define the content of the reference tag so a bit is added to the CDB that indicates if the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG is to used or not.

## Changes to SPC-4

### 1.0.1 Extended INQUIRY Data VPD page

The Extended INQUIRY Data VPD page (see table 1) provides the application client with a means to obtain information about the logical unit.

**Table 1 — Extended INQUIRY Data VPD page**

| Bit<br>Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | PERIPHERAL QUALIFIER | | | PERIPHERAL DEVICE TYPE | | | | |
| 1 | PAGE CODE (86h) | | | | | | | |
| 2 | Reserved | | | | | | | |
| 3 | PAGE LENGTH (3Ch) | | | | | | | |
| 4 | Reserved | | RTO | | | GRD_CHK | APP_CHK | REF_CHK |
| 5 | Reserved | | | GROUP_SUP | PRIOR_SUP | HEADSUP | ORDSUP | SIMPSUP |
| 6 | Reserved | | | | | | NV_SUP | V_SUP |
| 7 | Reserved | | | | | | | |
| 63 | | | | | | | | |

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are as defined in 6.4.2.

The PAGE LENGTH field specifies the length of the following VPD page data and shall be set to 60. The relationship between the PAGE LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.4.6.

~~A reference tag ownership (RTO) bit set to zero indicates that the logical unit does not support application client ownership of the LOGICAL BLOCK REFERENCE TAG field in the protection information (see SBC-2), if any. A RTO bit set to one indicates that the logical unit supports application client ownership of the LOGICAL BLOCK REFERENCE TAG field.~~

A reference tag ownership (RTO) field (see table 2) indicates if the logical unit supports application client ownership of the LOGICAL BLOCK REFERENCE TAG field in the protection information (see SBC-3), if any.

**Table 2 — RTO field**

| Code | Definition |
|---|---|
| 000b | Indicates that the logical unit does not support application client ownership of the LOGICAL BLOCK REFERENCE TAG field in the protection information (see SBC-3). |
| 001b | Indicates that the logical unit supports use by the application client of the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB (see SBC-3). |
| 010b | Reserved |
| 011b | Indicates that the logical unit supports application client ownership of the LOGICAL BLOCK REFERENCE TAG field (see SBC-3) and, if selected, the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB. |
| 100b - 111b | Reserved |

## Changes to SBC-3

## 1.1 Protection information model

### 1.1.1 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. 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-3).

~~For commands that are using protection information, the data-in buffer and/or data-out buffer shall consist of logical blocks with both user data and protection information. For commands that are not using protection information, the data-in buffer and/or data-out buffer shall consist of logical blocks with only user data.~~

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-3), then checking of the logical block reference tag within the service delivery subsystem without accounting for modified data pointers and data alignments may cause false errors when logical blocks are transmitted out of order.

### 1.1.2 Command processing restrictions

If the logical unit is formatted with protection information all commands in which the CDB contains a zero value in the RDPROTECT field, WRPROTECT field, and VRPROTECT field are valid. For those commands, the data-in buffer and/or data-out buffer associated with those commands shall consist of logical blocks with only user data.

If the logical unit is formatted with protection information not all commands in which the CDB contains a non-zero value in the RDPROTECT field, WRPROTECT field, and VRPROTECT field are valid. Commands that are allowed is indicated be the RTO_EN field returned in the READ CAPACITY (16) parameter data (see 1.6.2) as specified in table 3. For allowed commands using protection information, the data-in buffer and/or data-out buffer shall consist of logical blocks with both user data and protection information.

**Table 3 — RTO_EN field protection information command processing restrictions**

| RTO_EN **field** [a] | **Description** |
|---|---|
| 000b | For any command in which the CDB contains an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB 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. |
| 001b | For any command in which the CDB does not contain:<br>a) an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field; or<br>b) a RDPROTECT field set to zero, WRPROTECT field set to zero, or VRPROTECT field set to zero,<br>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.<br>For any command in which the CDB contains an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field with the XLB_INVALID bit set to one 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. |
| 010b | No restrictions placed on any commands as a result of a logical unit being formatted with protection information. |
| 011b - 111b | Reserved |

[a] Specified in the READ CAPACITY (16) parameter data (see 1.6.2).

Editor's Note 1: I have received requests to eliminate the XLB_INVALID bit from the 32-byte CDBs. This can be done in one of two ways. Option one would be to make 32-byte commands illegal when formatted in the new mode. This is reasonable as the new mode effectively eliminates much of the function used in the 32-byte commands. Option two would be to allow the 32-byte commands but ignore the logical block application tag mask field and the expected logical block application tag field when formatted in the new mode.

### 1.1.3 Protection information format

Table 4 defines the placement of protection information in a logical block.

**Table 4 — User data and protection information format**

| 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) |

The 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 LOGICAL BLOCK GUARD field contains the CRC (see 4.16.3) of the contents of the USER DATA field.

The LOGICAL BLOCK APPLICATION TAG field is set by the application client. A LOGICAL BLOCK APPLICATION TAG field set to FFFFh disables checking of all protection information for the logical block when reading from the medium. Otherwise, the contents of the logical block application tag 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-3). The contents of the LOGICAL BLOCK APPLICATION TAG field shall not be used to generate or check the CRC contained in the LOGICAL BLOCK GUARD field.

Editor's Note 2: The above change as accepted in 05-101r1.

~~The LOGICAL BLOCK REFERENCE TAG field is an incrementing value associated with the logical block. The LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer depends on the command being processed:~~

~~a) for a command that does not include an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field (e.g., READ (16)) the LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer shall contain the least significant four bytes of the LBA contained in the LOGICAL BLOCK ADDRESS field of the command; and~~

~~b) for a command that does include an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field (e.g., READ (32)) the LOGICAL BLOCK REFERENCE TAG field of the first logical block shall contain the value in the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field of the command. These commands are only processed if the medium was formatted with application client ownership of the logical block reference tag (i.e., with the RTO_REQ bit set to one in the FORMAT UNIT command (see 5.2)).~~

~~Each subsequent logical block in the data-in buffer and/or data-out buffer shall contain a logical block reference tag field with the logical block reference tag of the previous logical block plus one.~~

The LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer shall be set as specified in table 5.

**Table 5 — Setting the LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer**

| RTO_EN **field** [a] | **Description** |
|---|---|
| 000b | The content of the LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer is set to the least significant four bytes of the LBA contained in the LOGICAL BLOCK ADDRESS field of the command. |
| 001b | For any command in which the CDB contains an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field with the XLB_INVALID bit is set to zero the LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer shall contain the value in the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field of the command. |
| 010b | For any command in which the CDB contains:<br>a) does not contain an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field; or<br>b) contains an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field with the XLB_INVALID bit is set to one,<br>the contents of the LOGICAL BLOCK REFERENCE TAG field of the first the logical block in the data-in buffer and/or data-out buffer is vendor specific.<br>For any command in which the CDB contains an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field with the XLB_INVALID bit is set to zero the LOGICAL BLOCK REFERENCE TAG field of the first logical block in the data-in buffer and/or data-out buffer shall contain the value in the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field of the command. |
| 011b - 111b | Reserved |

[a] Specified in the READ CAPACITY (16) parameter data (see 1.6.2).

The LOGICAL BLOCK REFERENCE TAG field subsequent logical blocks in the data-in buffer and/or data-out buffer shall be set as specified in table 6.

**Table 6 — Setting the LOGICAL BLOCK REFERENCE TAG field of the subsequent logical blocks in the data-in buffer and/or data-out buffer**

| RTO_EN **field** [a] | **Description** |
|---|---|
| 000b | The content of the LOGICAL BLOCK REFERENCE TAG field of each subsequent logical block in the data-in buffer and/or data-out buffer shall contain the logical block reference tag of the previous logical block plus one. |
| 001b | For any command in which the CDB contains an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field with the XLB_INVALID bit is set to zero the content of the LOGICAL BLOCK REFERENCE TAG field of each subsequent logical block in the data-in buffer and/or data-out buffer shall contain the logical block reference tag of the previous logical block plus one. |
| 010b | For any command in which the CDB contains:<br>a) does not contain an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field; or<br>b) contains an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field with the XLB_INVALID bit is set to one,<br>the content of the LOGICAL BLOCK REFERENCE TAG field of subsequent logical blocks in the data-in buffer and/or data-out buffer is vendor specific.<br>For any command in which the CDB contains an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field with the XLB_INVALID bit is set to zero the content of the LOGICAL BLOCK REFERENCE TAG field of each subsequent logical block in the data-in buffer and/or data-out buffer shall contain the logical block reference tag of the previous logical block plus one. |
| 011b - 111b | Reserved |

[a] Specified in the READ CAPACITY (16) parameter data (see 1.6.2).

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

## 1.2 FORMAT UNIT command

### 1.2.1 FORMAT UNIT command overview

...

A format protection information (FMTPINFO) bit (see table 9) specifies if the device server enables or disables the use of protection information. ~~set to zero specifies that the device server shall disable the use of protection information (see 4.16) and format the medium to the block length specified in the mode parameter block descriptor of the mode parameter header (see SPC-3). A FMTPINFO bit set to one specifies that the device server shall enable the use of protection information (see 4.16) and format the medium to the block length specified in the mode parameter block descriptor of the mode parameter header plus eight (e.g., if the block length is 512, then the formatted block length is 520).~~ Following a successful format, the RTO_EN ~~bit~~ field in the READ CAPACITY (16) parameter data (see 1.6.2) indicates whether protection information (see 4.16) is enabled.

The reference tag own request (RTO_REQ) bit (see table 9) specifies whether the application client or the device server has ownership of the LOGICAL BLOCK REFERENCE TAG field in protection information (see 4.16.2). ~~If the FMTPINFO bit is set to one, and the RTO_REQ bit is set to one, the device server shall enable application client ownership of the LOGICAL BLOCK REFERENCE TAG field. If the FMTPINFO bit set to one and the RTO_REQ bit is set to zero, the device server shall disable application client ownership (i.e., enable device server ownership) of the LOGICAL BLOCK REFERENCE TAG field. If the FMTPINFO bit is set to zero and the RTO_REQ bit is set to one 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.~~ Following a successful format, the RTO_EN bit field in the READ CAPACITY (16) parameter data (see 1.6.2) indicates if the application client or the device server owns the LOGICAL BLOCK REFERENCE TAG field.

When protection information is written during a FORMAT UNIT command (i.e., the FMTPINFO bit is set to one) protection information shall be written to a default value of FFFFFFFF_FFFFFFFFh.

....

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

**Table 7 — Short parameter list header**

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

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

**Table 8 — Long parameter list header**

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

The PROTECTION FIELD USAGE field (see table 9) specifies the requested application tag and reference tag usage when the application client has ownership of the LOGICAL BLOCK REFERENCE TAG field in protection information (see 4.16.2).

**Table 9 — FMTPINFO bit, RTO_REQ bit, and PROTECTION FIELD USAGE field**

| RTO [b] | PROTECT [a] | FMTPINFO | RTO_REQ | PROTECTION FIELD USAGE | **Description** |
|---|---|---|---|---|---|
| xxxb | 0 | 0 [d] | 0 | 000b | The device server shall disable the use of protection information, if any (see 1.1). |
| xxxb | 0 | 0 | 0 | >000b | Illegal [g] |
| xxxb | 0 | 0 | 1 | xxxb | Illegal [f] |
| xxxb | 0 | 1 | x | xxxb | Illegal [f] |
| xxxb | 1 | 0 [d] | 0 | 000b | The device server shall disable the use of protection information, if any (see 1.1). |
| xxxb | 1 | 0 | 0 | >000b | Illegal [g] |
| xxxb | 1 | 0 | 1 | xxxb | Illegal [f] |
| 000b<br>001b<br>011b | 1 | 1 [c] [e] | 0 | 000b | The device server shall disable application client ownership (i.e., enable device server ownership) of the LOGICAL BLOCK REFERENCE TAG field in the CDB (see 1.1.3). |
| 000b<br>001b<br>011b | 1 | 1 | 0 | >000b | Illegal [g] |
| 000b | 1 | 1 | 1 | xxxb | Illegal [f] |
| 001b<br>011b | 1 | 1 [c] [e] | 1 | 000b | The logical unit shall be formatted to allow use by the application client of the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB (see 1.1.3). |
| 001b | 1 | 1 | 1 | 001b | Illegal [g] |
| 001b | 1 | 1 | 1 | 010b - 111b | Reserved |
| 011b | 1 | 1 [c] [e] | 1 | 001b | The logical unit shall be formatted to allow application client ownership of the LOGICAL BLOCK REFERENCE TAG field and, if selected, the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB (see 1.1.3). |
| 011b | 1 | 1 | 1 | 010b - 111b | Reserved |

[a] See the standard INQUIRY data (see SPC-3) for the definition of the PROTECT bit.

[b] See the Extended INQUIRY Data VPD page (see SPC-3) for the definition of the RTO field.

[c] The device server shall enable the use of protection information (see 4.16)

[d] The device server shall format the medium to the block length specified in the mode parameter block descriptor of the mode parameter header (see SPC-3).

[e] The device server shall format the medium to the block length specified in the mode parameter block descriptor of the mode parameter header plus eight (e.g., if the block length is 512, then the formatted block length is 520). Following a successful format, the PROT_EN bit in the READ CAPACITY (16) parameter data (see 1.6.2) indicates whether protection information (see 4.16) is enabled.

[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 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.

A format options valid (FOV) bit set to zero specifies that the device server shall use its default settings for the DPRY, DCRT, STPF, and IP bits. If the FOV bit is set to zero, the application client shall set these bits to zero. If the FOV bit is set to zero and any of the other bits listed in this paragraph are not set to zero, 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.

## 1.3 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 10.

**Table 10 — Protection information checking for READ (6)**

| Logical unit formatted with protection information | Shall device server transmit protection information? | Field in protection information [f] | 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 [g] | 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 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 an error is reported, 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-3) for the definitions of the GRD_CHK bit, APP_CHK bit, and REF_CHK bit.

~~[e] If the device server detects a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, it shall not check any protection information in the associated logical block.~~

[f] If the device server detects a:

a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and the RTO_EN field is set to 000b or 001b in the READ CAPACITY (16) parameter data (see 1.6.2); or

b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF FFFFh, and the RTO_EN field is set to 010b,

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

[g] If the RTO_EN ~~bit~~ field is set to zero~~ in the READ CAPACITY (16) parameter data (see 1.6)~~, 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 the RTO_EN ~~bit~~ field is not set to zero ~~one~~, the device server checks the 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.

## 1.4 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 11.

**Table 11 — RDPROTECT field** (part 1 of 4)

| Code | Logical unit formatted with protection information | Shall device server transmit protection information? | Field in protection information [i] | Extended INQUIRY Data VPD page bit value [g] | If check fails [d] [f], additional sense code |
|---|---|---|---|---|---|
| 000b[~~j~~] | 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 [k] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | | REF_CHK = 0 | No check performed |
| | No | | No protection information available to check | | |
| 001b<br>101b [b][~~j~~] | 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 [k] | 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 11 — RDPROTECT field** (part 2 of 4)

| Code | Logical unit formatted with protection information | Shall device server transmit protection information? | Field in protection information [i] | Extended INQUIRY Data VPD page bit value [g] | If check fails [d] [f], additional sense code |
|---|---|---|---|---|---|
| 010b [b]~~[j]~~ | 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 [k] | 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]~~[j]~~ | 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 | | | |

**Table 11 — RDPROTECT field** (part 3 of 4)

| Code | Logical unit formatted with protection information | Shall device server transmit protection information? | Field in protection information [i] | Extended INQUIRY Data VPD page bit value [g] | If check fails [d] [f], additional sense code |
|---|---|---|---|---|---|
| 100b [b] ~~[j]~~ | 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 - 111b | Reserved | | | | |

Table 11 — RDPROTECT field (part 4 of 4)

| Code | Logical unit formatted with protection information | Shall device server transmit protection information? | Field in protection information [i] | 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 4.16) 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.

[b] 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.

[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. If the READ (32) command (see 1.5) is used and the ATO bit is set to one in the Control mode page (see SPC-3), 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.

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

[e] Transmit protection information to the data-in buffer.

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

[g] See the Extended INQUIRY Data VPD page (see SPC-3) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bit.

~~[h] If the application client or device server detects a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, the checking of all protection information in the associated logical block shall be disabled.~~

[i] If the application client or device server detects a:

a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and the RTO_EN field is set to 000b or 001b in the READ CAPACITY (16) parameter data (see 1.6.2); or

b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF FFFFh, and the RTO_EN field is set to 010b,

then the checking of all protection information in the associated logical block shall be disabled.

~~[j] If the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 1.6), the device server may process the command. If the RTO_EN bit is set to one, READ (10) commands, READ (12) commands, and READ (16) commands with the RDPROTECT field set to 000b may be processed by the device server. If the RTO_EN bit is set to one, the device server shall terminate READ (10) commands, READ (12) commands, and READ (16) commands with the RDPROTECT field not set to 000b with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE.~~

[k] If the RTO_EN ~~bit~~ field is set to zero in the READ CAPACITY (16) parameter data (see 1.6.2), 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 the RTO_EN bit is set to one the device server checks the logical block reference tag based on the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB (see 4.16.2).~~ If the RTO_EN ~~bit~~ field is not set to zero ~~one~~ the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. This knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a READ (32) command (see 1.5) or by a method not defined by this standard.

## 1.5 READ (32) command

The READ (32) command (see table 12) requests that the device server read the specified logical block(s) and transfer them to the data-in buffer. Each logical block read includes user data and, if the medium is formatted with protection information enabled, protection information. Each logical block transferred includes user data and may include protection information, based on the RDPROTECT field and the medium format.

~~If the RTO_EN bit field is set to zero in the READ CAPACITY (16) parameter data (see 1.6.2), 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 COMMAND OPERATION CODE. If the RTO_EN bit field is not set to zero one, the device server may process the command (see 1.1.3)~~.

Table 12 — READ (32) command

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (7Fh) | | | | | | | |
| 1 | CONTROL | | | | | | | |
| 2 | Reserved | | | | | | | |
| 5 | | | | | | | | |
| 6 | Reserved | | | GROUP NUMBER | | | | |
| 7 | ADDITIONAL CDB LENGTH (18h) | | | | | | | |
| 8 | (MSB) | SERVICE ACTION (0009h) | | | | | | |
| 9 | | | | | | | | (LSB) |
| 10 | RDPROTECT | | | DPO | FUA | Reserved | FUA_NV | Reserved |
| 11 | Reserved | | | | | | | XLB_INVALID |
| 12 | (MSB) | LOGICAL BLOCK ADDRESS | | | | | | |
| 19 | | | | | | | | (LSB) |
| 20 | (MSB) | EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG | | | | | | |
| 23 | | | | | | | | (LSB) |
| 24 | (MSB) | EXPECTED LOGICAL BLOCK APPLICATION TAG | | | | | | |
| 25 | | | | | | | | (LSB) |
| 26 | (MSB) | LOGICAL BLOCK APPLICATION TAG MASK | | | | | | |
| 27 | | | | | | | | (LSB) |
| 28 | (MSB) | TRANSFER LENGTH | | | | | | |
| 31 | | | | | | | | (LSB) |

See the READ (10) command (see 1.4) for the definitions of the GROUP NUMBER field, the RDPROTECT field, the DPO bit, the FUA bit, the FUA_NV bit, the LOGICAL BLOCK ADDRESS field, and the TRANSFER LENGTH field.

An expected logical block invalid (XLB_INVALID) bit set to zero specifies that the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field, EXPECTED LOGICAL BLOCK APPLICATION TAG field, and LOGICAL BLOCK APPLICATION TAG MASK field are valid. An XLB_INVALID bit set to one specifies that the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field shall not be used when checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 11 in 1.4). In this case the value expected in the LOGICAL BLOCK REFERENCE TAG field is not defined by this standard.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 11 in 1.4) and the XLB_INVALID bit is set to zero, the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.16.2).

If the ATO bit is set to one in the Control mode page (see SPC-3) the XLB_INVALID bit is set to zero, and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 11 in 1.4), 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 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 the protection information.

The LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored if:

a) the XLB_INVALID bit is set to one;
b) the ATO bit is set to zero; or
c) the ATO bit is set to one in the Control mode page (see SPC-3) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 11 in 1.4).

## 1.6 READ CAPACITY (16) command

### 1.6.1 READ CAPACITY (16) command overview

The READ CAPACITY (16) command (see table 13) requests that the device server transfer parameter data describing the capacity and medium format of the direct-access block device to the data-in buffer. This command is mandatory if the logical unit supports protection information (see 4.16) and optional otherwise. This command is implemented as a service action of the SERVICE ACTION IN operation code (see A.2). This command may be processed as if it has a HEAD OF QUEUE task attribute (see 4.11).

**Table 13 — READ CAPACITY (16) command**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (9Eh) | | | | | | | |
| 1 | Reserved | | | SERVICE ACTION (10h) | | | | |
| 2 | (MSB) | LOGICAL BLOCK ADDRESS | | | | | | |
| 9 | | | | | | | | (LSB) |
| 10 | (MSB) | ALLOCATION LENGTH | | | | | | |
| 13 | | | | | | | | (LSB) |
| 14 | Reserved | | | | | | | PMI |
| 15 | CONTROL | | | | | | | |

See the READ CAPACITY (10) command (see 5.10) for definitions of the LOGICAL BLOCK ADDRESS field and the PMI bit.

The ALLOCATION LENGTH field specifies the maximum number of bytes that the application client has allocated for returned parameter data. An allocation length of zero indicates that no data shall be transferred. This condition shall not be considered as an error. The device server shall terminate transfers to the data-in buffer when the number of bytes specified by the ALLOCATION LENGTH field have been transferred or when all available data has been transferred, whichever is less. The contents of the parameter data shall not be altered to reflect the truncation, if any, that results from an insufficient allocation length.

### 1.6.2 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 4.6.

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

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | (MSB) | RETURNED LOGICAL BLOCK ADDRESS | | | | | | |
| 7 | | | | | | | | (LSB) |
| 8 | (MSB) | BLOCK LENGTH IN BYTES | | | | | | |
| 11 | | | | | | | | (LSB) |
| 12 | Reserved | | | | RTO_EN | | | PROT_EN |
| 13 | Reserved | | | | | | | |
| 31 | | | | | | | | |

The RETURNED LOGICAL BLOCK ADDRESS field and 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 5.10). The maximum value that shall be returned in the RETURNED LOGICAL BLOCK ADDRESS field is FFFFFFFF_FFFFFFFEh.

~~A reference tag own enable (RTO_EN) bit set to one indicates that application client ownership of the LOGICAL BLOCK REFERENCE TAG field in protection information is enabled (i.e., the medium was formatted with protection information (see 4.16) enabled and the RTO_REQ bit was set to one). An RTO_EN bit set to zero indicates that application client ownership of the LOGICAL BLOCK REFERENCE TAG field in protection information is disabled.~~

The reference tag owner (RTO_EN) field (see table 15) specifies the logical unit's current allowed application client usage of the LOGICAL BLOCK REFERENCE TAG field in protection information (see 4.16.2).

**Table 15 — RTO_EN field**

| Code | Description |
|---|---|
| 000b | Application client ownership of the LOGICAL BLOCK REFERENCE TAG field in protection information is disabled. |
| 001b | The logical unit only supports CDBs (see 1.1.3) that contain:<br>a) an EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field; and<br>b) a RDPROTECT field set to zero, WRPROTECT field set to zero, or VRPROTECT field set to zero. |
| 010b | The logical unit supports application client ownership of the LOGICAL BLOCK REFERENCE TAG field and, if selected, the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB (see 1.1.3). |
| 011b - 111b | Reserved |

A PROT_EN bit set to one indicates that the medium was formatted with protection information (see 4.16) enabled. A PROT_EN bit set to zero indicates that the medium was not formatted with protection information enabled.

## 1.7 VERIFY (10) command

...

If the BYTCHK bit is set to zero, the device server shall check the protection information read from the medium based on the VRPROTECT field as described in table 16.

**Table 16 — VRPROTECT field with BYTCHK set to zero - checking protection information read from the medium** (part 1 of 4)

| Code | Logical unit formatted with protection information | Field in protection information [h] | 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 [i] | 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. | | |
| 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 [i] | 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 [i] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | | | REF_CHK = 0 | No check performed |
| | No | Error condition [a] | | |

**Table 16 — VRPROTECT field with BYTCHK set to zero - checking protection information read from the medium** (part 2 of 4)

| Code | Logical unit formatted with protection information | Field in protection information [h] | 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] | | |

**Table 16 — VRPROTECT field with BYTCHK set to zero - checking protection information read from the medium** (part 3 of 4)

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

Table 16 — VRPROTECT field with BYTCHK set to zero - checking protection information read from the medium (part 4 of 4)

| Code | Logical unit formatted with protection information | Field in protection information [h] | 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 4.16) 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.

[b] 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.

[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. If the VERIFY (32) command (see 1.8) is used and the ATO bit is set to one in the Control mode page (see SPC-3), 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 an error is reported, the sense key shall be set to ABORTED COMMAND.

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

[f] See the Extended INQUIRY Data VPD page (see SPC-3) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bits.

~~[g] If the application client or device server detects a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, the checking of all protection information shall be disabled for the associated logical block.~~

[h] If the application client or device server detects a:

a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and the RTO_EN field is set to 000b or 001b in the READ CAPACITY (16) parameter data (see 1.6.2); or

b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF FFFFh, and the RTO_EN field is set to 010b,

then the checking of all protection information in the associated logical block shall be disabled.

[i] If the RTO_EN ~~bit~~ field is set to zero ~~in the READ CAPACITY (16) parameter data (see 1.6) (i.e., the command is a VERIFY (10) command, a VERIFY (12) command, or a VERIFY (16) command)~~, the device server ~~shall check~~ checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. I~~f the RTO_EN bit is set to one (i.e., the command is a VERIFY (32) command), the device server shall check the logical block reference tag based on the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB (see 4.16.2)~~. If the RTO_EN ~~bit~~ field is not set to zero ~~one~~ the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. This knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 1.8) or by a method not defined by this standard.

If the BYTCHK bit is set to one, 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 one - checking protection information read from the medium** (part 1 of 2)

| Code | Logical unit formatted with protection information | Field in protection information [h] | 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 [i] | 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] | | |

**Table 17 — 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 [h] | Extended INQUIRY Data VPD page bit value [f] | If check fails [d] [e], additional sense code |
|---|---|---|---|---|
| 110b - 111b | Reserved | | | |

[a] A verify operation to a logical unit that supports protection information (see 4.16) 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.

[b] 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.

[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. If the VERIFY (32) command (see 1.8) is used and the ATO bit is set to one in the Control mode page (see SPC-3), 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 an error is reported, the sense key shall be set to ABORTED COMMAND.

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

[f] See the Extended INQUIRY Data VPD page (see SPC-3) for the definitions of the GRD_CHK bit, the APP_CHK bit, and the REF_CHK bit.

~~[g] If the application client or device server detects a LOGICAL BLOCK APPLICATION TAG field set to FFFFh, the checking of all protection information shall be disabled for the associated logical block.~~

[h] If the application client or device server detects a:

a) LOGICAL BLOCK APPLICATION TAG field set to FFFFh and the RTO_EN field is set to 000b or 001b in the READ CAPACITY (16) parameter data (see 1.6.2); or

b) LOGICAL BLOCK APPLICATION TAG field set to FFFFh, LOGICAL BLOCK REFERENCE TAG field set to FFFF FFFFh, and the RTO_EN field is set to 010b,

then the checking of all protection information in the associated logical block shall be disabled.

[i] If the RTO_EN ~~bit~~ field is set to zero ~~in the READ CAPACITY (16) parameter data (see 1.6) (i.e., the command is a VERIFY (10) command, a VERIFY (12) command, or a VERIFY (16) command)~~, the device server ~~shall check~~ checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. I~~f the RTO_EN bit is set to one (i.e., the command is a VERIFY (32) command), the device server shall check the logical block reference tag based on the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB (see 4.16.2)~~. If the RTO_EN ~~bit~~ field is not set to zero ~~one~~ the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. This knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 1.8) or by a method not defined by this standard.

If the BYTCHK bit is set to one, the device server shall check the protection information transferred from the data-out buffer based on the VRPROTECT field as described in table 18.

**Table 18 — 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 [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 18 — 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 - 111b | Reserved | | | |

[a] A verify operation to a logical unit that supports protection information (see 4.16) 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.

[b] 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.

[c] 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-3) and if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the VERIFY (32) command (see 1.8) is used, 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 an error is reported, the sense key shall be set to ABORTED COMMAND.

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

[f] If the RTO_EN ~~bit~~ field is set to zero in the READ CAPACITY (16) parameter data (see 1.6.2)~~(i.e., the command is a VERIFY (10) command, a VERIFY (12) command, or a VERIFY (16) command)~~, the device server ~~shall check~~ checks the logical block reference tag by comparing it to the lower 4 bytes of the LBA associated with the logical block. ~~If the RTO_EN bit is set to one (i.e., the command is a VERIFY (32) command), the device server shall check checks the logical block reference tag based on the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB (see 4.16.2).~~ If the RTO_EN ~~bit~~ field is not set to zero ~~one~~ the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. This knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a VERIFY (32) command (see 1.8) or by a method not defined by this standard.

If the BYTCHK bit is set to one, 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 19.

**Table 19 — 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<br>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 | Shall | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No | Error condition [a] | | |

**Table 19 — 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 |
|---|---|---|---|---|
| 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 | Shall | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | 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 - 111b | Reserved | | | |

[a] A verify operation to a logical unit that supports protection information (see 4.16) 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.

[b] 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.

[c] If an error is reported, 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-3), 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-3), the logical block application tag may be modified by a device server.

## 1.8 VERIFY (32) command

The VERIFY (32) command (see table 20) requests that the device server verify the specified logical block(s) on the medium. Each logical block includes user data and may include protection information, based on the VRPROTECT field and the medium format.

~~If the RTO_EN bit field is set to zero in the READ CAPACITY (16) parameter data (see 1.6.2), the device server shall terminate this command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST~~

~~and the additional sense code set to INVALID COMMAND OPERATION CODE. If the RTO_EN bit field is not set to zero one, the device server may process the command (see 1.1.3)~~.

**Table 20 — VERIFY (32) command**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (7Fh) | | | | | | | |
| 1 | CONTROL | | | | | | | |
| 2 | Reserved | | | | | | | |
| 5 | | | | | | | | |
| 6 | Reserved | | | GROUP NUMBER | | | | |
| 7 | ADDITIONAL CDB LENGTH (18h) | | | | | | | |
| 8 | (MSB) | SERVICE ACTION (000Ah) | | | | | | |
| 9 | | | | | | | | (LSB) |
| 10 | VRPROTECT | | | DPO | Reserved | | BYTCHK | Reserved |
| 11 | Reserved | | | | | | | XLB_INVALID |
| 12 | (MSB) | LOGICAL BLOCK ADDRESS | | | | | | |
| 19 | | | | | | | | (LSB) |
| 20 | (MSB) | EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG | | | | | | |
| 23 | | | | | | | | (LSB) |
| 24 | (MSB) | EXPECTED LOGICAL BLOCK APPLICATION TAG | | | | | | |
| 25 | | | | | | | | (LSB) |
| 26 | (MSB) | LOGICAL BLOCK APPLICATION TAG MASK | | | | | | |
| 27 | | | | | | | | (LSB) |
| 28 | (MSB) | VERIFICATION LENGTH | | | | | | |
| 31 | | | | | | | | (LSB) |

See the VERIFY (10) command (see 1.7) for the definitions of the GROUP NUMBER field, VRPROTECT field, DPO bit, BYTCHK bit, LOGICAL BLOCK ADDRESS field, and VERIFICATION LENGTH field.

An expected logical block invalid (XLB_INVALID) bit set to zero specifies that the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field, EXPECTED LOGICAL BLOCK APPLICATION TAG field, and LOGICAL BLOCK APPLICATION TAG MASK field are valid. An XLB_INVALID bit set to one specifies that the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field shall not be used when checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 16, table 17, table 18, and table 19 in 1.7). In this case the value expected in the LOGICAL BLOCK REFERENCE TAG field is not defined by this standard.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 16, table 17, table 18, and table 19 in 1.7) and the XLB_INVALID bit is set to zero, the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.16.2).

If the ATO bit is set to one in the Control mode page (see SPC-3), the XLB_INVALID bit is set to zero, and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 16, table 17, table 18, and table 19 in 1.7), 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 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.

The LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored if:

a) the XLB_INVALID bit is set to one;
b) the ATO bit is set to zero; or
c) the ATO bit is set to one in the Control mode page (see SPC-3) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 16, table 17, table 18, and table 19 in 1.7).

## 1.9 WRITE (6) command

The WRITE (6) command (see table 21) requests that the device server transfer the specified logical block(s) from the data-out buffer and write them. Each logical block transferred includes user data but does not include protection information. Each logical block written includes user data and, if the medium is formatted with protection information enabled, protection information.

**Table 21 — WRITE (6) command**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (0Ah) | | | | | | | |
| 1 | Reserved | | | (MSB) | | | | |
| 2 | LOGICAL BLOCK ADDRESS | | | | | | | |
| 3 | | | | | | | | (LSB) |
| 4 | TRANSFER LENGTH | | | | | | | |
| 5 | CONTROL | | | | | | | |

The cache control bits are not provided for this command. Direct-access block devices with cache may have values for the cache control bits that may affect the WRITE (6) command, however no default value is defined by this standard. If explicit control is required, the WRITE (10) command should be used.

See the PRE-FETCH (10) command (see 5.3) for the definition of the LOGICAL BLOCK ADDRESS field.

The TRANSFER LENGTH field specifies the number of contiguous logical blocks of data that shall be transferred from the data-out buffer and written, starting with the logical block specified by the LOGICAL BLOCK ADDRESS field. A TRANSFER LENGTH field set to zero specifies that 256 logical blocks shall be written. Any other value specifies the number of logical blocks that shall be written. 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).

NOTE 1 - For the WRITE (10) command, WRITE (12) command, WRITE (16) command, and WRITE (32) command, a TRANSFER LENGTH field set to zero specifies that no logical blocks are transferred.

If a WRITE (6) command is received after protection information is enabled the device server shall set the protection information (see 4.16) as follows as it writes each logical block to the medium:

a) the LOGICAL BLOCK GUARD field set to a properly generated CRC (see 4.16.3);
b) the LOGICAL BLOCK REFERENCE TAG field set to:
   A) the least significant four bytes of the LBA, if the RTO_EN ~~bit~~ field is set to zero in the READ CAPACITY (16) parameter data (see 1.6.2); or
   B) FFFFFFFFh, if the RTO_EN ~~bit~~ field is not set to zero ~~one~~;

   and

c) the LOGICAL BLOCK APPLICATION TAG field set to:

A) FFFFh, if the ATO bit is set to one in the Control mode page (see SPC-3); or
B) any value, if the ATO bit is set to zero in the Control mode page (see SPC-3).

## 1.10 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 22.

**Table 22 — 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 | | |
| 001b [b] ~~j~~ | 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 [k] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No [a] | No protection information available to check | | |
| 010b [b] ~~j~~ | 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 [k] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No [a] | No protection information available to check | | |
| 011b [b] ~~j~~ | 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 | | |

**Table 22 — 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 |
|---|---|---|---|---|
| 100b [b] ~~[j]~~ | 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] ~~[j]~~ | 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 [k] | LOGICAL BLOCK REFERENCE TAG CHECK FAILED |
| | No [a] | No protection information available to check | | |

Table 22 — 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 |
|---|---|---|---|---|
| 110b - 111b | Reserved | | | |

[a] A write operation to a logical unit that supports protection information (see 4.16) 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.

[b] 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.

[c] 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-3) and if it has knowledge of the contents of the LOGICAL BLOCK APPLICATION TAG field. If the WRITE (32) command (see 1.11) is used, 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 an error is reported, 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 4.16.3.2) into each LOGICAL BLOCK GUARD field.

[g] If the RTO_EN ~~bit~~ field is set to zero in the READ CAPACITY (16) parameter data (see 1.6.2), 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 RTO_EN ~~bit~~ field is not set to zero ~~one~~, the device server shall write a value of FFFFFFFFh into the 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-3), the device server shall write FFFFh into each LOGICAL BLOCK APPLICATION TAG field. If the ATO bit is set to zero, 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 the RTO_EN bit is set to zero in the READ CAPACITY (16) parameter data (see 1.6), the device server may process the command. If the RTO_EN bit is set to one, WRITE (10) commands, WRITE (12) commands, and WRITE (16) commands with the WRPROTECT field set to 000b may be processed by the device server. If the RTO_EN bit is set to one, the device server shall terminate WRITE (10) commands, WRITE (12) commands, and WRITE (16) commands with the WRPROTECT field not set to 000b with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE.~~

[k] If the RTO_EN ~~bit~~ field is set to zero ~~in the READ CAPACITY (16) parameter data (see 1.6.1)~~, 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 the RTO_EN bit is set to one (i.e., the command is a WRITE (32) command), the device server checks the logical block reference tag based on the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in the CDB (see 4.16.2)~~. If the RTO_EN ~~bit~~ field is not set to zero the device server checks the logical block reference tag if it has knowledge of the contents of the LOGICAL BLOCK REFERENCE TAG field. This knowledge may be acquired through the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field in a WRITE (32) command (see 1.11) or by a method not defined by this standard.

## 1.11 WRITE (32) command

The WRITE (32) command (see table 23) requests that the device server transfer the specified logical block(s) from the data-out buffer and write them. Each logical block transferred includes user data and may include protection information, based on the WRPROTECT field and the medium format. Each logical block written includes user data and, if the medium is formatted with protection information enabled, protection information.

~~If the RTO_EN bit field is set to zero in the READ CAPACITY (16) parameter data (see 1.6.2), the device server shall terminate the WRITE (32) command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE. If the RTO_EN bit field is not set to zero one, the device server may process the command (see 1.1.3)~~.

**Table 23 — WRITE (32) command**

| Byte\Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| 0 | OPERATION CODE (7Fh) | | | | | | | |
| 1 | CONTROL | | | | | | | |
| 2 | Reserved | | | | | | | |
| 5 | | | | | | | | |
| 6 | Reserved | | | GROUP NUMBER | | | | |
| 7 | ADDITIONAL CDB LENGTH (18h) | | | | | | | |
| 8 | (MSB) | SERVICE ACTION (000Bh) | | | | | | |
| 9 | | | | | | | | (LSB) |
| 10 | WRPROTECT | | | DPO | FUA | Reserved | FUA_NV | Reserved |
| 11 | Reserved | | | | | | | XLB_INVALID |
| 12 | (MSB) | LOGICAL BLOCK ADDRESS | | | | | | |
| 19 | | | | | | | | (LSB) |
| 20 | (MSB) | EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG | | | | | | |
| 23 | | | | | | | | (LSB) |
| 24 | (MSB) | EXPECTED LOGICAL BLOCK APPLICATION TAG | | | | | | |
| 25 | | | | | | | | (LSB) |
| 26 | (MSB) | LOGICAL BLOCK APPLICATION TAG MASK | | | | | | |
| 27 | | | | | | | | (LSB) |
| 28 | (MSB) | TRANSFER LENGTH | | | | | | |
| 31 | | | | | | | | (LSB) |

See the WRITE (10) command (see 1.10) for the definitions of the GROUP NUMBER field, the WRPROTECT field, the DPO bit, the FUA bit, the FUA_NV bit, the LOGICAL BLOCK ADDRESS field, and the TRANSFER LENGTH field.

An expected logical block invalid (XLB_INVALID) bit set to zero specifies that the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field, EXPECTED LOGICAL BLOCK APPLICATION TAG field, and LOGICAL BLOCK APPLICATION TAG MASK field are valid. An XLB_INVALID bit set to one specifies that the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field shall not be used when checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 22 in 1.10). In this case the value expected in the LOGICAL BLOCK REFERENCE TAG field is not defined by this standard.

When checking of the LOGICAL BLOCK REFERENCE TAG field is enabled (see table 22 in 1.10) and the XLB_INVALID bit is set to zero, the EXPECTED INITIAL LOGICAL BLOCK REFERENCE TAG field contains the value of the LOGICAL BLOCK REFERENCE TAG field expected in the protection information of the first logical block accessed by the command instead of a value based on the LBA (see 4.16.2).

If the ATO bit is set to one in the Control mode page (see SPC-3), the XLB_INVALID bit is set to zero, and checking of the LOGICAL BLOCK APPLICATION TAG field is enabled (see table 22 in 1.10), 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 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.

The LOGICAL BLOCK APPLICATION TAG MASK field and the EXPECTED LOGICAL BLOCK APPLICATION TAG field shall be ignored if:

a) the XLB_INVALID bit is set to one;
b) the ATO bit is set to zero; or
c) the ATO bit is set to one in the Control mode page (see SPC-3) and checking of the LOGICAL BLOCK APPLICATION TAG field is disabled (see table 22 in 1.10).