Date: October 20, 2003

To: T10 Committee (SCSI)

From: George Penokie (IBM/Tivoli)

Subject: End-to-End Data Protection

1 Overview

Any inconsistencies between this section and the remaining sections in the proposal should be ignored in this section as the remaining sections are what will be placed into the relevant standards. This section is only here as a brief description of the overall proposal and should not be considered normative.

There is an need (real or imagined) for a standardized end-to-end data protection mechanism to be defined. The logical place to such a definition is the SCSI command and architecture standards, as most storage uses SCSI commands to read/write data to and from storage devices. What follows is a proposal that provides a set of SCSI tools that will enable end-to-end data protection. This set of SCSI tools are defined to accomplish this goal:

a) without interfering with existing proprietary methods;
b) with a minimum of options; and
c) by defining minimal changes to CDBs while maintaining backward compatibility.

The set of SCSI tools will consist of the following:

a) Three level data protection on each data block transferred across the interconnect that consists of;
    A) A 2-byte CRC that covers the user data of the data block. The CRC is generated at or before the application client and preserved at the logical unit.
    B) A 4-byte incrementing LBA tag. The incrementing LBA tag is set by the application client during write operation to the value of the least significant 4 bytes of the write command's LBA field on the first data block transferred and incremented by one on each data block transferred until all the blocks for the command have been transferred. The increment LBA tag values for each data block that is read back shall be the same value that was received for that data block.
    C) A 2-byte application defined tag.
b) A bit in the non-Read Read CDBs (e.g, VERIFY) to allow a logical unit to return the protection information.
    A) If zero then do not transmit any protection information. The logical unit shall not check the contents of the protection information.
    B) If one then transmit the protection information. The logical unit may determine if the data block is valid by checking the contents of the protection information. If the logical unit determines there is a an error it shall generate a check condition. A read to a logical unit that has not been formatted to transmit the data protection information may fail with a check condition. In the case where the logical unit has not been formatted and does not check the bit the contents of the protection information is not defined by this standard and as a result should cause an error at the application client.
c) A two bit field in the READ commands (excluding the READ (6) command) that would control the reading and checking of protection information.
    A) If 00b then do not transmit any protection information. If the logical unit has been formatted with protection information the logical unit may determine if the data block is valid by checking the contents of the protection information. If the logical unit determines there is a an error it shall generate a check condition.
    B) If 01b then transmit the protection information. The logical unit may determine if the data block is valid by checking the contents of the protection information. If the logical unit determines there is a an error it shall generate a check condition. A read to a logical unit that has not been formatted to transmit the data protection information may fail with a check condition. In the case where the logical unit has not been formatted and does not check the bit the contents of the protection information is not defined by this standard and as a result should cause an error at the application client.

C) If 10b then transmit the protection information. The logical unit shall not check the contents of the DATA BLOCK GUARD field. The logical unit may determine if the data block is valid by checking the contents of the protection information except for the DATA BLOCK GUARD field. If the logical unit determines there is a an error it shall generate a check condition. A read to a logical unit that has not been formatted to transmit the data protection information may fail with a check condition. In the case where the logical unit has not been formatted and does not check the bit the contents of the protection information is not defined by this standard and as a result should cause an error at the application client.

D) If 11b then transmit the protection information. The logical unit shall not check the contents of the protection information. A read to a logical unit that has not been formatted to transmit the data protection information may fail with a check condition. In the case where the logical unit has not been formatted and does not check the bit the contents of the protection information is not defined by this standard and as a result should cause an error at the application client.

d) A two bit field in Write CDBs to allow the protection information to be written with no checks.
   - A) If 00h then preserve the contents of the protection information (e.g., write to media, store in non-volatile memory, recalculate on read back). The logical unit shall determine if the data block is valid by checking the contents of the protection information. If the logical unit determines there is a an error it shall generate a check condition. If the logical unit has not been formatted to accept protection information it shall generate a check condition.
   - B) If 01h then preserve the contents of the protection information (e.g., write to media, store in non-volatile memory, recalculate on read back). The logical unit shall not check the contents of the protection information. If the logical unit has not been formatted to accept protection information it shall generate a check condition.
   - C) If 10h then the contents of the protection information shall not be preserved. The logical unit shall determine if the data block is valid by checking the contents of the protection information. If the logical unit determines there is a an error it shall generate a check condition. In the case where the logical unit has not been formatted with protection information and does not check the CDB protection field the logical units response to the command is not defined by this standard.

e) A bit in the Format CDB to cause 8 bytes to be added to the block size of the logical unit being formatted.
   - A) If zero then format the medium to the block length defined in the mode parameter block descriptor of the Mode parameter header.
   - B) If one then format the medium to the block length defined in the mode parameter block descriptor of the Mode parameter header plus 8 (e.g., if block length = 512 the formatted block length is 520). The block length shall be a multiple of four. If the block length is not a multiple of four the logical unit shall generate a check condition.

f) All commands that request block length information (e.g., Read Capacity, Mode Sense) shall return the block size of the data excluding the eight bytes of protection information (e.g., a 520 byte data block on a device formatted with protection information returns 512 in the block length field).

g) A two bit field in the Standard Inquiry Data to indicate support of protection information.
   - A) If 00b then no protection is supported.
   - B) If 10b then protection is supported but not enabled
   - C) If 11b then protection is supported and enabled.

h) A bit in a mode page that forces the logical unit to write to media the contents of the data block guard field.
   - A) If zero then the contents of the data block guard field shall be preserved and may be written to media.
   - B) If one then the contents of the data block guard field shall be written to media.

SBC-2 additions

3.1 Definitions

3.1.1 default protection information: Values placed into the protection information fields if an application client does not indicate specific protection information values (see 4.0.2).

3.1.2 protection information: Fields appended to each block of data that contain a cyclic redundancy check (CRC), a data block application tag, and a data block reference tag.

4.x Protection information model (new section)

4.0.1 Protection information overview

This data protection model provides for the protection of the data while it is being transferred between a sender and a receiver. The protection information is generated at the application layer and may be checked by any object along the I_T_L nexus. Once received, the protection information is retained (e.g., write to media, store in non-volatile memory, recalculate on read back) by the device server until overwritten (e.g., power loss and reset events have no effect on the retention of protection information).

4.0.2 Protection information format

See figure 1 for the placement of the protection information.

.

Table 1 — Protection information format

Byte\Bit	7	6	5	4	3	2	1	0
0	DATA BLOCK							
n								
n + 1	(MSB) DATA BLOCK GUARD							
n + 2								(LSB)
n + 3	(MSB) DATA BLOCK APPLICATION TAG							
n + 4								(LSB)
n + 5	(MSB) DATA BLOCK REFERENCE TAG							
n + 8								(LSB)

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

The DATA BLOCK GUARD field contains the CRC (see 4.0.3) of the contents of the DATA BLOCK field. The default value for the DATA BLOCK field shall be a properly generated CRC (see 4.0.3)

The DATA BLOCK APPLICATION TAG field is set by the application client. The contents of the data block application tag are not defined by this standard. The DATA BLOCK APPLICATION TAG field may be modified by a device server if the APP_TAG_OWN bit is set to zero (see 4.0.45). If the APP_TAG_OWN bit is set to one the default value for the DATA BLOCK APPLICATION TAG field shall be FFFFh. The contents of the DATA BLOCK APPLICATION TAG field shall not be used to generate or check the CRC contained in the DATA BLOCK GUARD field.

The DATA BLOCK REFERENCE TAG field is set to the least significant four bytes of the logical block address to which the data block is associated. The first data block transmitted shall contain the least significant four bytes of the logical block address contained in the LOGICAL BLOCK ADDRESS field of the command associated with the data being transferred. Each subsequent data block's DATA BLOCK REFERENCE TAG field shall contain the data block reference tag of the previous data block plus one. The default value for the DATA BLOCK REFERENCE TAG field shall be the least significant four bytes of the LBA of the data block being written or formatted. The contents of the DATA BLOCK REFERENCE TAG field shall not be used to generate or check the CRC contained in the DATA BLOCK GUARD field.

4.0.3 Data block guard protection

If data protection is enabled, the application client shall append a data block guard to each block of data to be transmitted. The data block guard shall contain a CRC that is generated from the contents of the DATA BLOCK field.

Table 2 defines the CRC polynomials.

Table 2 — CRC polynomials

Function	Definition
F(x)	A polynomial of degree k-1 that is used to represent the k bits of the data block covered by the CRC. For the purposes of the CRC, the coefficient of the highest order term shall be the first transmitted bit of the DATA BLOCK field.
G(x)	The generator polynomial: $G(x) = x^{16} + x^{15} + x^{11} + x^9 + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$ (i.e., G(x) = 18BB7h)
R(x)	The remainder polynomial, which is of degree less than 16.
P(x)	The remainder polynomial on the receive checking side, which is of degree less than 16. A value of P(0) indicates no error was detected.
Q(x)	The greatest multiple of G(x) in $(x^{16} \times F(x)) + (x^k \times L(x))$
Q'(x)	$x^{16} \times Q(x)$
M(x)	The sequence that is transmitted.
M'(x)	The sequence that is received.

4.0.4 CRC generation

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

CRC value in data block = R(x)

The CRC is calculated by the following equation:

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

The following equation specifies that the CRC is appended to the end of F(x):

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

The bit order of F(x) presented to the CRC function is MSB to LSB two bytes at a time until the contents of the DATA BLOCK field are all processed. This order is shown in figure 1.

Editor's Note 1: Need to remove the MSB and LSB wording above as the data block field has no MSB or LSB. Of change the data block field in table 1 to have a MSB and LSB.

.

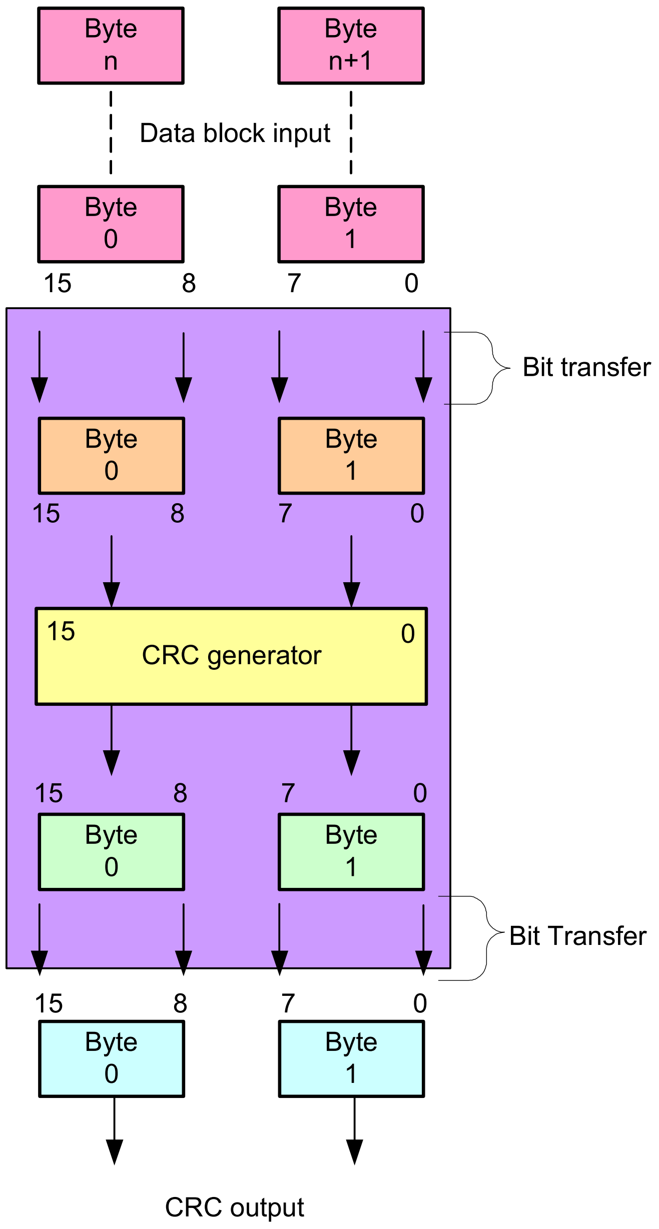

Figure 1 — CRC generator bit order

Editor's Note 2: Received the following comment: Figure 1 the "Bit transfer" portions are unnecessary. Get rid of the purple, orange, and green boxes and just feed into the yellow CRC generator.

4.0.5 CRC checking

The received sequence M'(x) may differ from the transmitted sequence M(x) if there are transmission errors. The process of checking the sequence for validity involves dividing the received sequence by G(x) and testing the remainder. Mathematically, the received checking is shown by the following equation:

$$x^{16} \times \frac{M'(x)}{G(x)} = Q'(x) + \frac{P(x)}{G(x)}$$

In the absence of errors remainder P(x) is zero.

The bit order of F(x) presented to the CRC checking function is the same order as the CRC generation bit order (see figure 1).

4.0.6 Test cases

Using the polynomial described in table 2, the CRC calculated for a 32-byte transfer of all 00h is: 0000h,

Using the polynomial described in table 2, the CRC calculated for a 32-byte transfer of all FFh: A293h.

Using the polynomial described in table 2, the CRC calculated for a 32-byte transfer of an incrementing pattern from 00h to 1Fh is: 0224h.

Using the polynomial described in table 2, the CRC calculated for a 32-byte transfer of FFFFh followed by 30 bytes 00h: 21B8h

Using the polynomial described in table 2, the CRC calculated for a 32-byte transfer of a decrementing pattern from FFh to E0h: A0B7h

4.0.7 Application of protected data

Before an application client transmits or receives protected data it shall:

1) Determine if a logical unit supports protected data using the INQUIRY command (see SPC-3);
2) If protected data is supported then determine if the logical unit is formatted to accept protected information using the INQUIRY command;
3) If the logical unit supports protected information and is not formatted to accept protected information then the application client formats the logical unit with protected information usage enabled; and
4) If the logical unit supports protected information and is formatted to accept protected information then the application client may use read commands that support protected information and should use write commands that support protected information.

4.0.8 Protected data commands

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

The commands that are affected when protection information is enabled are:

a) FORMAT UNIT;
b) READ (6)/(10)/(12)/(16);
c) READ LONG;
d) REASSIGN BLOCKS;
e) REBUILD (16)/(32);
f) REGENERATE (16)/(32);
g) SYNCHRONIZE CACHE (10)/(16);
h) VERIFY (10)/(12)/(16);
i) WRITE (6)/(10)/(12)/(16);
j) WRITE AND VERIFY (10)/(12)/(16);
k) WRITE LONG;
l) WRITE SAME (10)/(12);
m) XDREAD (10)/(32);
n) XDWRITE (10)/(32);
o) XDWRITE EXTENDED (16)/(32)/(64);
p) XDWRITEREAD (10)/(32); and
q) XPWRITE (10)/(32).

Editor's Note 3: The above list could be moved to the commands table with a new yes/no column indicating which are affected by protection information.

If a WRITE (6) command is received after protection information is enabled the device server shall insert default protection information (see 3.1.1) after each data block before writing the data block to the medium. If the device server has been formatted with protection information and is not capable of inserting default protection information it shall terminate the command with a CHECK CONDITION status and the sense data shall be set to MEDIUM ERROR with the additional sense code set to CANNOT WRITE MEDIUM - UNKNOWN FORMAT.

A READ (6) command may be sent to a logical unit that has protection information enabled, however, only the requested data blocks, excluding the protection information, shall be returned.

Commands that result in the return of the length in bytes of each logical block (e.g., MODE SENSE, READ CAPACITY) shall return the length of the user data and shall not include the length of the protection information (e.g., if the user data plus the protection information is equal to 520 bytes then 512 is returned).

Editor's Note 4: When putting this into SBC-2 there needs to be references to this section placed into the WRITE(6), READ(6), READ CAPACITY, and any other command the comment in the last paragraph applies to. In the case of mode pages or headers there should be also be a reference to this section.

4.0.9 FORMAT UNIT command

4.0.9.1 FORMAT UNIT command overview

The FORMAT UNIT command (see table 3) formats the medium into application client addressable logical blocks per the application client defined options. In addition, the medium may be certified and control structures may be created for the management of the medium and defects. The degree that the medium is altered by this command is vendor-specific.

Table 3 — FORMAT UNIT command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (04h)							
1	FMTPINFO	Reserved	LONGLIST	FMTDATA	CMPLIST	DEFECT LIST FORMAT		
2	Vendor specific							
3	(MSB)	INTERLEAVE						
4								(LSB)
5	CONTROL							

A format protection information (FMTPINFO) bit of zero indicates that the device server shall format the medium to the block length specified in the mode parameter block descriptor of the Mode parameter header (see x.x.x). A FMTPINFO bit of one indicates the device server shall format the medium block length specified in the mode parameter block descriptor of the Mode parameter header plus eight (e.g., if the block length equals 512 the formatted block length is 520). A successful format with protection information shall cause the PROTECT field in the standard INQUIRY data (see SPC-3) to be changed resulting in a unit attention condition.

...

An initialization pattern (IP) bit of zero indicates that an initialization pattern descriptor is not included and that the device server shall use its default initialization pattern. An IP bit of one indicates that an initialization pattern descriptor (see 4.0.9.2) is included in the FORMAT UNIT parameter list immediately following the defect list header. If the FMTPINFO bit and the IP bit are set to one then the DATA BLOCK GUARD field shall be set to a properly generated CRC (see 4.0.3), and DATA BLOCK REFERENCE TAG field shall be set to the LBA of the data block being formatted (see 4.0.2)

If the FMTPINFO bit, the IP bit, APP_TAG_OWN bit (see 4.0.45), and the DB_APP_TAG bit are set to one then the DATA BLOCK APPLICATION TAG field shall be set to the application tag initialization pattern (see 4.0.9.2). If the FMTPINFO bit, the IP bit, and the DB_APP_TAG bit are set to one and APP_TAG_OWN bit is set to zero then the DATA BLOCK APPLICATION TAG field may be set to the application tag initialization pattern.

4.0.9.2 Initialization pattern option

The initialization pattern option specifies that the logical blocks contain the specified initialization pattern. The initialization pattern descriptor (see table 4) is sent to the device server as part of the FORMAT UNIT parameter list.

Table 4 — Initialization pattern descriptor

Byte\Bit	7	6	5	4	3	2	1	0
0	IP MODIFIER		SI	DB_APP_TAG	Reserved			
1	PATTERN TYPE							
2	(MSB)	INITIALIZATION PATTERN LENGTH						
3								(LSB)
4		INITIALIZATION PATTERN						
n								
n+1	(MSB)	APPLICATION TAG INITIALIZATION PATTERN						
n+2								(LSB)

A data block application tag (DB_APP_TAG) bit set to zero indicates there is no APPLICATION TAG INITIALIZATION PATTERN field within the initialization pattern descriptor and that a value of FFFFh shall be placed into the DATA BLOCK APPLICATION TAG field. A DB_APP_TAG bit set to one indicates there is an APPLICATION TAG INITIALIZATION PATTERN field in the last two bytes of the initialization pattern descriptor.

If present, the APPLICATION TAG INITIALIZATION PATTERN field contains the value to be placed into the protection information DATA BLOCK APPLICATION TAG field of each data block.

4.0.10 READ (10) command

The READ (10) command (see table 5) requests that the device server transfer data to the application client. The most recent data value written in the addressed logical block shall be returned.

Table 5 — READ (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (28h)							
1	Reserved	RDPROTECT		DPO	FUA	Reserved		RELADR
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved							
7	(MSB)	TRANSFER LENGTH						
8								(LSB)
9	CONTROL							

The RDPROTECT field is defined in table 6.

Table 6 — RDPROTECT field

Value	Logical unit formatted with protection information	Transmit	Device server protection information validity checking rules [f]		
			Field	Checked	If check fails [d] Additional sense code
00b	Yes	No	DATA BLOCK GUARD	May	DATA BLOCK GUARD CHECK FAILED
			DATA BLOCK APPLICATION TAG	May [c]	DATA BLOCK APPLICATION TAG CHECK FAILED
			DATA BLOCK REFERENCE TAG	May	DATA BLOCK REFERENCE TAG CHECK FAILED
	No		No protection information available to check		
01b [b]	Yes	Yes [e]	DATA BLOCK GUARD	May	DATA BLOCK GUARD CHECK FAILED
			DATA BLOCK APPLICATION TAG	May [c]	DATA BLOCK APPLICATION TAG CHECK FAILED
			DATA BLOCK REFERENCE TAG	May	DATA BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to transmit to the application client or for checking			
10b [b]	Yes	Yes [e]	DATA BLOCK GUARD	Shall not	No check performed
			DATA BLOCK APPLICATION TAG	May [c]	DATA BLOCK APPLICATION TAG CHECK FAILED
			DATA BLOCK REFERENCE TAG	May	DATA BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to transmit to the application client or for checking			
11b [b]	Yes	Yes [e]	DATA BLOCK GUARD	Shall not	No check performed
			DATA BLOCK APPLICATION TAG	Shall not	No check performed
			DATA BLOCK REFERENCE TAG	Shall not	No check performed
	No [a]	No protection information available to transmit to the application client or for checking			

[a] A read operation to a logical unit that supports protection information (see table 36) and has not been formatted with protection information shall fail with a CHECK CONDITION status. The sense data shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.
[b] If the logical unit does not support protection information the requested command should fail, however, the specific error failure is not defined by this standard.
[c] The device server may check the data block application tag if it has knowledge of the contents of the DATA BLOCK APPLICATION TAG field. The method for acquiring this knowledge is not defined by this standard.
[d] If an error is reported the sense key shall be set to ABORTED COMMAND.
[e] Transmit the protection information to the application client.
[f] If multiple errors occur, the selection of which error to report is not defined by this standard.

4.0.11 READ (12) command

The READ (12) command (see table 7) requests that the device server transfer data to the application client from the medium.

Table 7 — READ (12) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (A8h)							
1	Reserved	RDPROTECT		DPO	FUA	Reserved		RELADR
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	(MSB)	TRANSFER LENGTH						
9								(LSB)
10	Reserved							
11	CONTROL							

4.0.12 READ (16) command

The READ (16) command (see table 8) requests that the device server transfer data to the application client.

Table 8 — READ (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (88h)							
1	Reserved	RDPROTECT		DPO	FUA	Reserved		RELADR
2	(MSB)	LOGICAL BLOCK ADDRESS						
9								(LSB)
10	(MSB)	TRANSFER LENGTH						
13								(LSB)
14	Reserved							
15	CONTROL							

4.0.13 READ LONG command

The READ LONG command (see table 41) requests that the device server transfer data to the application client. The data passed during the READ LONG command is vendor-specific, but shall include the data bytes, any protection information, and the ECC bytes recorded on the medium. The most recent data written, or to be written, in the addressed logical block shall be returned. READ LONG is independent of the Read-Write Error Recovery mode page but does allow retries.

4.0.14 REASSIGN BLOCKS command

The REASSIGN BLOCKS command (see table 42) requests the device server to reassign the defective logical blocks and that logical block's protection information to another area on the medium set aside for this purpose. The device server should also record the location of the defective logical blocks to the grown defect list if such a list is supported. More than one physical or logical block may be relocated by each defect descriptor sent by the application client. This command does not alter the contents of the PLIST (see 4.0.9).

4.0.15 REBUILD (16) Command

The REBUILD (16) command (see table 46) requests that the target write to the medium the XOR data generated from the specified source devices. The target, acting as a temporary initiator, issues READ commands to retrieve the specified data and protection information, if any. READ (10) should be used for accesses to SCSI devices supporting less than 2 Terabytes, and READ (16) should be used for accesses to SCSI devices supporting greater than or equal to 2 Terabytes

4.0.16 REBUILD (32) Command

The REBUILD (32) command (see table 50) requests that the target write to the medium the XOR data generated from the specified source devices. The target, acting as a temporary initiator, issues READ commands to retrieve the specified data and protection information, if any.

4.0.17 REGENERATE (16) command

The REGENERATE (16) command (see table 53) requests that the target write to the buffer the XOR data generated from its own medium and the specified source devices. The target, acting as a temporary initiator, issues READ commands to retrieve the specified data and protection information, if any. The resulting XOR data is retained in the target's buffer until it is retrieved by an XDREAD command with a starting LOGICAL BLOCK ADDRESS and TRANSFER LENGTH that match, or are a subset of, the LOGICAL BLOCK ADDRESS and REGENERATE LENGTH of this command.

4.0.18 REGENERATE (32) command

The REGENERATE (32) command (see table 54) requests that the target write to the buffer the XOR data generated from its own medium and the specified source devices. The target, acting as a temporary initiator, issues READ commands to retrieve the specified data and protection information, if any.

4.0.19 SYNCHRONIZE CACHE (10) command

The SYNCHRONIZE CACHE (10) command (see table 60) ensures that logical blocks in the cache memory, within the specified range, have their most recent data value and protection information, if any, recorded on the physical medium. If a more recent data value for a logical block within the specified range exists in the cache memory than on the physical medium, then the logical block from the cache memory shall be written to the physical medium. Logical blocks may not be removed from the cache memory as a result of the synchronize cache operation. The synchronize cache function is also required implicitly by other SCSI functions as defined in other clauses of this standard.

4.0.20 SYNCHRONIZE CACHE (16) command

The SYNCHRONIZE CACHE (16) command (see table 61) ensures that logical blocks in the cache memory, within the specified range, have their most recent data value and protection information, if any, recorded on the physical medium. If a more recent data value for a logical block within the specified range exists in the cache memory than on the physical medium, then the logical block from the cache memory shall be written to the physical medium. Logical blocks may not be removed from the cache memory as a result of the synchronize cache operation. The synchronize cache function is also required implicitly by other SCSI functions as defined in other clauses of this standard

4.0.21 VERIFY (10) command

The VERIFY (10) command (see table 9) requests that the device server verify the data written on the medium.

Table 9 — VERIFY (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Fh)							
1	Reserved	VRPROTECT		DPO	Reserved	BLKVFY	BYTCHK	RELADR
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Restricted for MMC-4	Reserved						
7	(MSB)	VERIFICATION LENGTH						
8								(LSB)
9	CONTROL							

See 4.2.1.8 for reservation requirements for this command. See 4.0.10 for a description of the cache control bit (DPO). See the LOCK UNLOCK CACHE (10) command (see 5.2.3) for a definition of the RELADR bit and the LOGICAL BLOCK ADDRESS field.

If the MODE SELECT command is implemented, and the Verify Error Recovery mode page is also implemented, then the current settings in that page specifies the verification criteria. If the Verify Error Recovery mode page is not implemented, then the verification criteria is vendor-specific.

If the byte check (BYTCHK) bit is zero, a medium verification shall be performed with no data comparison. For protection information comparison requirements when BYTCHK is set to zero see table 10. If the BYTCHK bit is one, a byte-by-byte comparison of data written on the medium and the data transferred from the application client shall be performed. For protection information comparison requirements when BYTCHK is set to one see table 11 for the protection information byte-by-byte comparison requirements, see table 12 for the protection information checking requirements on data transferred from the application client. and see table 13 for the protection information checking requirements on data written on the medium.

If the comparison is unsuccessful for any reason, the device server shall return CHECK CONDITION status and the sense key shall be set to MISCOMPARE with the appropriate additional sense code for the condition.

For direct access block devices, the blank verify (BLKVFY) bit shall be considered reserved. For optical and write-once block devices, the BLKVFY BIT is defined as follows. If the BLKVFY bit is zero, the device server shall not verify that the blocks are blank. If the BLKVFY bit is one, the device server shall verify that the blocks are blank. If the BYTCHK is one and the BLKVFY bit is one the device server shall return CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.

The VRPROTECT field with BYTCHK set to zero is defined in table 10.

Table 10 — VRPROTECT field with BYTCHK = 0

Value	Logical unit formatted with protection information	Device server protection information validity checking rules [e]		
		Field	Checked	If check fails [d] Additional sense code
00b	Yes	DATA BLOCK GUARD	May	DATA BLOCK GUARD CHECK FAILED
		DATA BLOCK APPLICATION TAG	May [c]	DATA BLOCK APPLICATION TAG CHECK FAILED
		DATA BLOCK REFERENCE TAG	May	DATA BLOCK REFERENCE TAG CHECK FAILED
	No	No protection information available to check		
01b [b]	Yes	DATA BLOCK GUARD	May	DATA BLOCK GUARD CHECK FAILED
		DATA BLOCK APPLICATION TAG	May [c]	DATA BLOCK APPLICATION TAG CHECK FAILED
		DATA BLOCK REFERENCE TAG	May	DATA BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to check		
10b [b]	Yes	DATA BLOCK GUARD	Shall not	No check performed
		DATA BLOCK APPLICATION TAG	May [c]	DATA BLOCK APPLICATION TAG CHECK FAILED
		DATA BLOCK REFERENCE TAG	May	DATA BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to check		
11b [b]	Yes	DATA BLOCK GUARD	Shall not	No check performed
		DATA BLOCK APPLICATION TAG	Shall not	No check performed
		DATA BLOCK REFERENCE TAG	Shall not	No check performed
	No [a]	No protection information available to check		

[a] A verify operation to a logical unit that supports protection information (see table 36) and has not been formatted with protection information shall fail with a CHECK CONDITION status. The sense data shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.
[b] If the logical unit does not support protection information the requested command should fail, however, the specific error failure is not defined by this standard.
[c] The device server may check the data block application tag if it has knowledge of the contents of the DATA BLOCK APPLICATION TAG field. The method for acquiring this knowledge is 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.

The protection information byte-by-byte comparison requirements with BYTCHK set to one is defined in table 11.

Table 11 — VRPROTECT field with BYTCHK = 1 (byte-by-byte comparison)

Value	Logical unit formatted with protection information	Device server protection information byte-by-byte comparison requirements [d]		
		Field	Checked	If check fails [c] Additional sense code
00b	Yes	No protection information received from application client to compare		
	No	No protection information received from application client to compare		
01b [b]	Yes	DATA BLOCK GUARD	Shall	DATA BLOCK GUARD CHECK FAILED
		DATA BLOCK APPLICATION TAG (APP_TAG_OWN = 1) [e]	Shall	DATA BLOCK APPLICATION TAG CHECK FAILED
		DATA BLOCK APPLICATION TAG (APP_TAG_OWN = 0) [f]	Shall not	No compare performed
		DATA BLOCK REFERENCE TAG	Shall	DATA BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to compare		
10b [b]	Yes	DATA BLOCK GUARD	Shall not	No compare performed
		DATA BLOCK APPLICATION TAG (APP_TAG_OWN = 1) [e]	Shall	DATA BLOCK APPLICATION TAG CHECK FAILED
		DATA BLOCK APPLICATION TAG (APP_TAG_OWN = 0) [f]	Shall not	No compare performed
		DATA BLOCK REFERENCE TAG	Shall	DATA BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to compare		
11b [b]	Yes	DATA BLOCK GUARD	Shall	DATA BLOCK GUARD CHECK FAILED
		DATA BLOCK APPLICATION TAG (APP_TAG_OWN = 1) [e]	Shall	DATA BLOCK APPLICATION TAG CHECK FAILED
		DATA BLOCK APPLICATION TAG (APP_TAG_OWN = 0) [f]	Shall not	No compare performed
		DATA BLOCK REFERENCE TAG	Shall	DATA BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to compare		

[a] A verify operation to a logical unit that supports protection information (see table 36) and has not been formatted with protection information shall fail with a CHECK CONDITION status. The sense data shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.
[b] If the logical unit does not support protection information the requested command should fail, however, the specific error failure is not defined by this standard.
[c] If an error is reported the sense key shall be set to MISSCOMPARE.
[d] If multiple errors occur, the selection of which error to report is not defined by this standard.
[e] The data block application tag shall not be modified by a device server.
[f] The data block application tag may be modified by a device server.

The application client protection information validity checking requirements with BYTCHK set to one is defined in table 12.

Table 12 — VRPROTECT field with BYTCHK = 1 (application client transferred checking)

Value	Logical unit formatted with protection information	Device server protection information validity checking requirements on protection information transferred from application client [e]		
		Field	Checked	If check fails [d] Additional sense code
00b	Yes	No protection information received from application client to check		
	No	No protection information received from application client to check		
01b [b]	Yes	DATA BLOCK GUARD	Shall	DATA BLOCK GUARD CHECK FAILED
		DATA BLOCK APPLICATION TAG	Shall not	No check performed
		DATA BLOCK REFERENCE TAG	Shall	DATA BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to check		
10b [b]	Yes	DATA BLOCK GUARD	Shall not	No check performed
		DATA BLOCK APPLICATION TAG	May [c]	DATA BLOCK APPLICATION TAG CHECK FAILED
		DATA BLOCK REFERENCE TAG	May	DATA BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to check		
11b [b]	Yes	DATA BLOCK GUARD	Shall not	No check performed
		DATA BLOCK APPLICATION TAG	Shall not	No check performed
		DATA BLOCK REFERENCE TAG	Shall not	No check performed
	No [a]	No protection information available to check		

[a] A verify operation to a logical unit that supports protection information (see table 36) and has not been formatted with protection information shall fail with a CHECK CONDITION status. The sense data shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.
[b] If the logical unit does not support protection information the requested command should fail, however, the specific error failure is not defined by this standard.
[c] The device server may check the data block application tag if it has knowledge of the contents of the DATA BLOCK APPLICATION TAG field. The method for acquiring this knowledge is 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.

The protection information validity checking requirements for protection information written on the medium with BYTCHK set to one is defined in table 13.

Table 13 — VRPROTECT field with BYTCHK = 1 (medium data block checking)

Value	Logical unit formatted with protection information	Device server protection information validity checking requirements on protection information written on the medium [d]		
		Field	Checked	If check fails [c]
				Additional sense code
00b	Yes	DATA BLOCK GUARD	May	DATA BLOCK GUARD CHECK FAILED
		DATA BLOCK APPLICATION TAG	May [c]	DATA BLOCK APPLICATION TAG CHECK FAILED
		DATA BLOCK REFERENCE TAG	May	DATA BLOCK REFERENCE TAG CHECK FAILED
	No	No protection information available to check		
01b [b]	Yes	DATA BLOCK GUARD	Shall not	No check performed
		DATA BLOCK APPLICATION TAG	Shall not	No check performed
		DATA BLOCK REFERENCE TAG	Shall not	No check performed
	No [a]	No protection information available to transmit to the application client or for checking		
10b [b]	Yes	DATA BLOCK GUARD	Shall not	No check performed
		DATA BLOCK APPLICATION TAG	Shall not	No check performed
		DATA BLOCK REFERENCE TAG	Shall not	No check performed
	No [a]	No protection information available to transmit to the application client or for checking		
11b [b]	Yes	DATA BLOCK GUARD	Shall not	No check performed
		DATA BLOCK APPLICATION TAG	Shall not	No check performed
		DATA BLOCK REFERENCE TAG	Shall not	No check performed
	No [a]	No protection information available to transmit to the application client or for checking		

[a] A verify operation to a logical unit that supports protection information (see table 36) and has not been formatted with protection information shall fail with a CHECK CONDITION status. The sense data shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.
[b] If the logical unit does not support protection information the requested command should fail, however, the specific error failure is not defined by this standard.
[c] If an error is reported the sense key shall be set to ABORTED COMMAND.
[d] If multiple errors occur, the selection of which error to report is not defined by this standard.

The VERIFICATION LENGTH field specifies the number of contiguous logical blocks of data or blanks that shall be verified. A VERIFICATION LENGTH of zero indicates that no logical blocks shall be verified. This condition shall not be considered as an error. Any other value indicates the number of logical blocks that shall be verified. The VERIFICATION LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.1.4.2).

4.0.22 VERIFY (12) command

The VERIFY (12) command (see table 14) requests that the device server verify the data on the medium.

Table 14 — VERIFY (12) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (AFh)							
1	Reserved	VRPROTECT		DPO	Reserved	BLKVFY	BYTCHK	RELADR
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	(MSB)	VERIFICATION LENGTH						
9								(LSB)
10	Reserved							
11	CONTROL							

See 4.2.1.8 for reservation requirements for this command. See the VERIFY (10) command (see 4.0.21) for a description of the fields in this command.

4.0.23 VERIFY (16) command

The VERIFY (16) command (see table 15) requests that the device server verify the data written on the medium.

Table 15 — VERIFY (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Fh)							
1	Reserved	VRPROTECT		DPO	Reserved	BLKVFY	BYTCHK	RELADR
2	(MSB)	LOGICAL BLOCK ADDRESS						
9								(LSB)
10	(MSB)	VERIFICATION LENGTH						
13								(LSB)
14	Reserved							
15	CONTROL							

See 4.2.1.8 for reservation requirements for this command. See the VERIFY (10) command (see 4.0.21) for a description of the fields in this command.

4.0.24 WRITE (10) command

The WRITE (10) command (see table 16) requests that the device server write the data transferred by the application client to the medium.

Table 16 — WRITE (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Ah)							
1	Reserved	WRPROTECT		DPO	FUA	EBP	Reserved	RELADR
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved							
7	(MSB)	TRANSFER LENGTH						
8								(LSB)
9	CONTROL							

The WRPROTECT field is defined in table 17.

Table 17 — WRPROTECT field

Value	Logical unit formatted with protection information	Device server protection information validity checking rules [g]		
		Field	Checked	If check fails [d] Additional sense code
00b	Yes [e f]	No protection information received from application client to check		
	No	No protection information received from application client to check		
01b [b]	Yes [e]	DATA BLOCK GUARD	Shall	DATA BLOCK GUARD CHECK FAILED
		DATA BLOCK APPLICATION TAG	Shall not	No check performed
		DATA BLOCK REFERENCE TAG	Shall	DATA BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to check		
10b [b]	Yes [e]	DATA BLOCK GUARD	Shall not	No check performed
		DATA BLOCK APPLICATION TAG	May [c]	DATA BLOCK APPLICATION TAG CHECK FAILED
		DATA BLOCK REFERENCE TAG	May	DATA BLOCK REFERENCE TAG CHECK FAILED
	No [a]	No protection information available to check		
11b [b]	Yes [e]	DATA BLOCK GUARD	Shall not	No check performed
		DATA BLOCK APPLICATION TAG	Shall not	No check performed
		DATA BLOCK REFERENCE TAG	Shall not	No check performed
	No [a]	No protection information available to check		

[a] A write operation to a logical unit that supports protection information (see table 36) and has not been formatted with protection information shall fail with a CHECK CONDITION status. The sense data shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB.
[b] If the logical unit does not support protection information the requested command should fail, however, the specific error failure is not defined by this standard.
[c] The device server may check the data block application tag if it has knowledge of the contents of the DATA BLOCK APPLICATION TAG field. The method for acquiring this knowledge is not defined by this standard.
[d] If an error is reported the sense key shall be set to ABORTED COMMAND.
[e] Preserve the contents of the protection information (e.g., write to media, store in non-volatile memory).
[f] The device server shall insert default protection information (see 4.0.1) after each data block before writing the data block to the medium.
[g] If multiple errors occur, the selection of which error to report is not defined by this standard.

4.0.25 WRITE (12) command

The WRITE (12) command (see table 18) requests that the device server write the data transferred from the application client to the medium.

Table 18 — WRITE (12) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (AAh)							
1	Reserved	WRPROTECT		DPO	FUA	Reserved		RELADR
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	(MSB)	TRANSFER LENGTH						
9								(LSB)
10	Restricted for MMC-4	Reserved						
11	CONTROL							

See 4.2.1.8 for reservation requirements for this command. See the WRITE (10) command (see 4.0.24) for a description of the fields in this command.

4.0.26 WRITE (16) command

The WRITE (16) command (see table 19) requests that the device server write the data transferred by the application client to the medium.

Table 19 — WRITE (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Ah)							
1	Reserved	WRPROTECT		DPO	FUA	Reserved		RELADR
2	(MSB)	LOGICAL BLOCK ADDRESS						
9								(LSB)
10	(MSB)	TRANSFER LENGTH						
13								(LSB)
14	Reserved							
15	CONTROL							

See 4.2.1.8 for reservation requirements for this command. See the WRITE (10) command (see 4.0.24) for a description of the fields in this command.

4.0.27 WRITE AND VERIFY (10) command

The WRITE AND VERIFY (10) command (see table 20) requests that the device server write the data transferred from the application client to the medium and then verify that the data and protection information, if any, is correctly written. The data is only transferred once from the application client to the device server.

Table 20 — WRITE AND VERIFY (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (2Eh)							
1	Reserved	WRPROTECT		DPO	Reserved	EBP	BYTCHK	RELADR
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved							
7	(MSB)	TRANSFER LENGTH						
8								(LSB)
9	CONTROL							

See 4.2.1.8 for reservation requirements for this command. See the LOCK UNLOCK CACHE (10) command (see 5.2.3) for a definition of the RELADR bit and the LOGICAL BLOCK ADDRESS field. See the WRITE (10) command (4.0.24) for a definition of the TRANSFER LENGTH field and the WRPROTECT field. See 4.0.10 for a description of the cache control bit (DPO). See the WRITE (10) command (see 4.0.24) for a description of the EBP bit.

If the MODE SELECT command is implemented, and the Verify Error Recovery mode page is also implemented, then the current settings in that mode page (along with the AWRE bit from the Read-Write Error Recovery mode page) specify the verification error criteria. If these mode pages are not implemented, then the verification criteria is vendor-specific.

A byte check (BYTCHK) bit of zero requests a medium verification to be performed with no data comparison. A BYTCHK bit of one requests a byte-by-byte comparison of data written on the medium and the data transferred from the application client. If the comparison is unsuccessful for any reason, the device server shall return CHECK CONDITION status with the sense key set to MISCOMPARE with the appropriate additional sense code for the condition. ~~If the APP_TAG_OWN bit is set to zero the device server shall not do a byte-by-byte comparison of the DATA BLOCK APPLICATION TAG field.~~

Editor's Note 5: The above should be deleted as the device server should be able to check whatever it wrote to the medium, it makes no difference what is written.

NOTE 1 - The WRITE AND VERIFY command specifically states that the data are not to be transferred twice (i.e., once for the write pass, and once for the verify pass) when performing a byte compare. If there is a need for two transfers to occur (e.g., to ensure the integrity of the path to the media), then the application client should issue a WRITE command with a LINK bit of one followed by a VERIFY command with a BYTCHK bit of one, transferring the same data on each command.

4.0.28 WRITE AND VERIFY (12) command

The WRITE AND VERIFY (12) command (see table 21) requests that the device server write the data transferred from the application client to the medium and then verify that the data and protection information, if any, is correctly written.

Table 21 — WRITE AND VERIFY(12) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (AEh)							
1	Reserved	WRPROTECT		DPO	Reserved	EBP	BYTCHK	RELADR
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	(MSB)	TRANSFER LENGTH						
9								(LSB)
10	Reserved							
11	CONTROL							

See 4.2.1.8 for reservation requirements for this command. See the WRITE AND VERIFY (10) command (see 4.0.27) for a description of the bits in this command.

4.0.29 WRITE AND VERIFY (16) command

The WRITE AND VERIFY (16) command (see table 22) requests that the device server write the data transferred from the application client to the medium and then verify that the data and protection information, if any, is correctly written. The data is only transferred once from the application client to the device server.

Table 22 — WRITE AND VERIFY (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (8Eh)							
1	Reserved	WRPROTECT		DPO	Reserved	EBP	BYTCHK	RELADR
2	(MSB)	LOGICAL BLOCK ADDRESS						
9								(LSB)
10	(MSB)	TRANSFER LENGTH						
13								(LSB)
14	Reserved							
15	CONTROL							

See 4.2.1.8 for reservation requirements for this command. See the WRITE AND VERIFY (10) command (see 4.0.27) for a description of the fields in this command.

4.0.30 WRITE LONG command

The WRITE LONG command (see table 72) requests that the device server write the data transferred by the application client to the medium. The data passed during the WRITE LONG command is implementation specific, but shall include the data bytes, any protection information, and the ECC bytes.

4.0.31 WRITE SAME (10) command

The WRITE SAME (10) command (see table 23) requests that the device server write the single block of data transferred by the application client to the medium multiple times to consecutive multiple logical blocks.

If the medium is formatted with protection information the value in the DATA BLOCK REFERENCE TAG field shall be placed into the DATA BLOCK REFERENCE TAG field (see 4.0.2) of the first logical block written to the medium. Into each of the following logical blocks the data block reference tag received in the data transferred by the application client incremented by one shall be placed into the data block reference tag of that data block (i.e., each data block written to the medium has a data block reference tag value of one greater than the previous data block). If the APP_TAG_OWN bit (see SPC-3) is set to one the data block application tag received in the single block of data shall be placed in the DATA BLOCK APPLICATION TAG field each logical block. If the APP_TAG_OWN bit (see SPC-3) is set to zero the data block application tag received in the single block of data may be placed in the DATA BLOCK APPLICATION TAG field each logical block.

Table 23 — WRITE SAME (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (41h)							
1	Reserved	WRPROTECT		Reserved		PBDATA	LBDATA	RELADR
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved							
7	(MSB)	NUMBER OF BLOCKS						
8								(LSB)
9	CONTROL							

See 4.2.1.8 for reservation requirements for this command. See the LOCK UNLOCK CACHE (10) command (see 5.2.3) for a definition of the RELADR bit and the LOGICAL BLOCK ADDRESS field. See the WRITE (10) command (4.0.24) for a definition of the WRPROTECT field.

4.0.32 WRITE SAME (16) command

The WRITE SAME (16) command (see table 24) requests that the device server write the single block of data transferred by the application client to the medium multiple times to consecutive multiple logical blocks.

If the medium is formatted with protection information the value in the DATA BLOCK REFERENCE TAG field shall be placed into the DATA BLOCK REFERENCE TAG field (see 4.0.2) of the first logical block written to the medium. Into each of the following logical blocks the data block reference tag received in the data transferred by the application client incremented by one shall be placed into the data block reference tag of that data block (i.e., each data block written to the medium has a data block reference tag value of one greater than the previous data block). If the APP_TAG_OWN bit (see 4.0.45) is set to zero the data block application tag received in the single block of data may be placed in the DATA BLOCK APPLICATION TAG field each logical block.

Table 24 — WRITE SAME (16) command

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

See 4.2.1.8 for reservation requirements for this command. See the WRITE SAME (10) command (see 4.0.31) for a description of the fields in this command.

4.0.33 XDREAD (10) command

The XDREAD (10) command (see table 25) requests that the target transfer to the initiator the XOR data generated by an XDWRITE or REGENERATE command.

Table 25 — XDREAD (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (52h)							
1	XORPINFO	Reserved						
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved							
7	(MSB)	TRANSFER LENGTH						
8								(LSB)
9	CONTROL							

See 4.2.1.8 for reservation requirements for this command.

If the XOR protection information (XORPINFO) is set to zero the protection information, if any, shall not be verified or transmitted. If the XORPINFO is set to one the protection information shall be transmitted but shall not be verified.

The XOR data transferred is identified by the LOGICAL BLOCK ADDRESS and TRANSFER LENGTH. The LOGICAL BLOCK ADDRESS and TRANSFER LENGTH shall be the same as, or a subset of, those specified in a prior XDWRITE or REGENERATE command. If a match is not found the command is terminated with a CHECK CONDITION status and the sense key shall be set to ILLEGAL REQUEST with the additional sense code set to INVALID FIELD IN CDB. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.1.4.2).

4.0.34 XDREAD (32) command

The XDREAD (32) command (see table 26) requests that the target transfer to the initiator the XOR data generated by an XDWRITE or REGENERATE command.

Table 26 — XDREAD (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
6								
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0003h)						
9								(LSB)
10	XORPINFO	Reserved						
11	Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS						
19								(LSB)
20	Reserved							
27								
28	(MSB)	TRANSFER LENGTH						
31								(LSB)

See 4.2.1.8 for reservation requirements for this command. See the XDREAD (10) command (see 4.0.33) and SPC-3 for a description of the fields in this command.

4.0.35 XDWRITE (10) command

The XDWRITE (10) command (see table 27) requests that the target XOR the data transferred including the protection information, if any, with the data on the medium. The resulting XOR data is stored by the target until it is retrieved by an XDREAD (10) command.

Table 27 — XDWRITE (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (50h)							
1	Reserved	WRPROTECT		DPO	FUA	DISABLE WRITE	Reserved	
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved							
7	(MSB)	TRANSFER LENGTH						
8								(LSB)
9	CONTROL							

See 4.2.1.8 for reservation requirements for this command. See the READ (10) command (see 4.0.10) for a definition of the cache control bits (DPO and FUA). See the WRITE (10) command (4.0.24) for a definition of the WRPROTECT field.

4.0.36 XDWRITE (32) command

The XDWRITE (32) command (see table 28) requests that the target XOR the data transferred including the protection information, if any, with the data on the medium. The resulting XOR data is stored by the target until it is retrieved by an XDREAD (32) command.

Table 28 — XDWRITE (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
6								
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0004h)						
9								(LSB)
10	Reserved	WRPROTECT		DPO	FUA	DISABLE WRITE	Reserved	
11	Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS						
19								(LSB)
20	Reserved							
27								
28	(MSB)	TRANSFER LENGTH						
31								(LSB)

See 4.2.1.8 for reservation requirements for this command. See the XDWRITE (10) command (see 4.0.35) and SPC-3 for a description of the fields in this command.

4.0.37 XDWRITEREAD (10) command

The XDWRITEREAD (10) command (see table 29) requests that the target XOR the data transferred (data-out) including the protection information, if any, with the data on the medium and return the resulting XOR data (data-in). This is the equivalent to an XDWRITE (10) followed by an XDREAD (10) with the same logical block address and transfer length. This command is only available on transport protocols supporting bidirectional commands.

Table 29 — XDWRITEREAD (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (53h)							
1	XORPINFO	WRPROTECT		DPO	FUA	DISABLE WRITE	Reserved	
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved							
7	(MSB)	TRANSFER LENGTH						
8								(LSB)
9	CONTROL							

See 4.2.1.8 for reservation requirements for this command. See the XWRITE (10) command (see 4.0.35) and XDREAD (10) command (see 4.0.33) for a description of the fields in this command.

4.0.38 XDWRITEREAD (32) command

The XDWRITEREAD (32) command (see table 30) requests that the target XOR the data transferred (data-out) including the protection information, if any, with the data on the medium and return the resulting XOR data (data-in). This is the equivalent to an XDWRITE (32) followed by an XDREAD (32) with the same logical block address and transfer length. This command is only available on transport protocols supporting bidirectional commands.

Table 30 — XDWRITEREAD (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
6								
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0007h)						
9								(LSB)
10	XORPINFO	WRPROTECT		DPO	FUA	DISABLE WRITE	Reserved	
11	Reserved							
12	(MSB)	LOGICAL BLOCK ADDRESS						
19								(LSB)
20	Reserved							
27								
28	(MSB)	TRANSFER LENGTH						
31								(LSB)

See 4.2.1.8 for reservation requirements for this command. See the XDWRITEREAD (10) command (see 4.0.37) and SPC-3 for a description of the fields in this command.

4.0.39 XDWRITE EXTENDED (16) command

The XDWRITE EXTENDED (16) command (see table 31) requests that the target XOR the data transferred including the protection information, if any, with the data on the medium. The resulting XOR data including the protection information, if any, may subsequently be sent to a secondary device using an XPWRITE (10) or XPWRITE (32) command. The target, acting as a temporary initiator, issues XPWRITE commands to retrieve the specified data. XPWRITE (16) should be used for access to SCSI devices supporting less than 2

Terabytes, and XPWRITE (32) should be used for accesses to SCSI devices supporting greater than or equal to 2 Terabytes.

Table 31 — XDWRITE EXTENDED (16) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (80h)							
1	Reserved	WRPROTECT		DPO	FUA	DISABLE WRITE	PORT CONTROL	
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	(MSB)	SECONDARY LOGICAL BLOCK ADDRESS						
9								(LSB)
10	(MSB)	TRANSFER LENGTH						
13								(LSB)
14	SECONDARY ADDRESS							
15	CONTROL							

See 4.2.1.8 for reservation requirements for this command. See 4.0.10 for a definition of the DPO and FUA bits. See the WRITE (10) command (4.0.24) for a definition of the WRPROTECT field.

4.0.40 XDWRITE EXTENDED (32) command

The XDWRITE EXTENDED (32) command (see table 32) requests that the target XOR the data transferred including the protection information, if any, with the data on the medium. The resulting XOR data may subsequently be sent to a secondary device using an XPWRITE (32) command.

Table 32 — XDWRITE EXTENDED (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
6								
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0005h)						
9								(LSB)
10	Reserved	WRPROTECT		DPO	FUA	DISABLE WRITE	PORT CONTROL	
11	SECONDARY ADDRESS							
12	(MSB)	LOGICAL BLOCK ADDRESS						
19								(LSB)
20	(MSB)	SECONDARY LOGICAL BLOCK ADDRESS						
27								(LSB)
28	(MSB)	TRANSFER LENGTH						
31								(LSB)

See 4.2.1.8 for reservation requirements for this command. See the XWRITE EXTENDED (16) command (see 4.0.39) and SPC-3 for a description of the fields in this command.

4.0.41 XDWRITE EXTENDED (64) command

The XDWRITE EXTENDED (64) command (see table 33) requests that the target XOR the data transferred including the protection information, if any, with the data on the medium. The resulting XOR data may subsequently be sent to a secondary device using an XPWRITE (32) command.

Table 33 — XDWRITE EXTENDED (64) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
6								
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0005h)						
9								(LSB)
10	Reserved	WRPROTECT		DPO	FUA	DISABLE WRITE	PORT CONTROL	
12	SECONDARY ADDRESS DESCRIPTOR							
43								
44	(MSB)	LOGICAL BLOCK ADDRESS						
51								(LSB)
52	(MSB)	SECONDARY LOGICAL BLOCK ADDRESS						
59								(LSB)
60	(MSB)	TRANSFER LENGTH						
63								(LSB)

See 4.2.1.8 for reservation requirements for this command.

The SECONDARY ADDRESS DESCRIPTOR field contains the logical unit identifier of the logical unit that will receive the XOR data transfer. The format of this field conforms to one of the target descriptor formats of the EXTENDED COPY command as specified in SPC-3.

See the XWRITE EXTENDED (16) command (see 4.0.39) and SPC-3 for a description of the other fields in this command.

4.0.42 XPWRITE (10) command

The XPWRITE (10) command (see table 34) requests that the target XOR the data transferred including the protection information, if any, with the data on the medium and then write the XOR data to the medium.

Table 34 — XPWRITE (10) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (51h)							
1	Reserved	WRPROTECT		DPO	FUA	Reserved		
2	(MSB)	LOGICAL BLOCK ADDRESS						
5								(LSB)
6	Reserved							
7	(MSB)	TRANSFER LENGTH						
8								(LSB)
9	CONTROL							

See 4.2.1.8 for reservation requirements for this command. See 4.0.10 for a definition of the DPO and FUA bits. See the WRITE (10) command (4.0.24) for a definition of the WRPROTECT field.

The LOGICAL BLOCK ADDRESS field specifies the starting logical block address where the target shall read data from its medium. It also specifies the starting logical block address where the XOR result data shall be written to the medium.

The TRANSFER LENGTH field specifies the number of blocks that shall be read from the medium. It also specifies the number of blocks that shall be written to the medium. The TRANSFER LENGTH field is constrained by the MAXIMUM TRANSFER LENGTH field in the Block Limits VPD page (see 6.1.4.2).

4.0.43 XPWRITE (32) command

The XPWRITE (32) command (see table 35) requests that the target XOR the data transferred including the protection information, if any, with the data on the medium and then write the XOR data to the medium.

Table 35 — XPWRITE (32) command

Byte\Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
6								
7	ADDITIONAL CDB LENGTH (18h)							
8	(MSB)	SERVICE ACTION (0006h)						
9								(LSB)
10	Reserved	WRPROTECT		DPO	FUA	Reserved		
44	(MSB)	LOGICAL BLOCK ADDRESS						
51								(LSB)
52	Reserved							
59								
60	(MSB)	TRANSFER LENGTH						
63								(LSB)

See 4.2.1.8 for reservation requirements for this command. See the XPWRITE (10) command (see 4.0.42) and SPC-3 for a description of the fields in this command.

SPC-3 additions

4.0.44 Standard INQUIRY data

The standard INQUIRY data (see table 36) shall contain at least 36 bytes.

Table 36 — Standard INQUIRY data format

Byte \ Bit	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	RMB	Reserved						
2	VERSION							
3	Obsolete	Obsolete	NORMACA	HISUP	RESPONSE DATA FORMAT			
4	ADDITIONAL LENGTH (n-4)							
5	SCCS	ACC	ALUA		3PC	Reserved	PROTECT	
6	BQUE	ENCSERV	VS	MULTIP	MCHNGR	Obsolete	Obsolete	ADDR16[a]
7	RELADR	Obsolete	WBUS16[a]	SYNC[a]	LINKED	Obsolete	CMDQUE	VS
8	(MSB)	VENDOR IDENTIFICATION						
15								(LSB)
16	(MSB)	PRODUCT IDENTIFICATION						
31								(LSB)
32	(MSB)	PRODUCT REVISION LEVEL						
35								(LSB)
36	Vendor specific							
55								
56	Reserved				CLOCKING[a]		QAS[a]	IUS[a]
57	Reserved							
58	(MSB)	VERSION DESCRIPTOR 1						
59								(LSB)
	⋮							
72	(MSB)	VERSION DESCRIPTOR 8						
73								(LSB)
74	Reserved							
95								
	Vendor specific parameters							
96	Vendor specific							
n								

[a] The meanings of these fields are specific to SPI-5 (see 6.4.3). For SCSI protocols other than the SCSI Parallel Interface, these fields are reserved.

The PROTECT field is defined in table 37

Table 37 — Peripheral qualifier

Qualifier	Description
00b	This logical unit does not support the protection information (see SBC-2).
01b	This logical unit supports the protection information but the medium has not been formatted to include protection information.
10b	Reserved
11b	The medium for this logical unit has been has been formatted to include protection information with each logical block.

4.0.45 Control mode page

The Control mode page (see table 38) provides controls over several SCSI features that are applicable to all device types such as tagged queuing and error logging.

Table 38 — Control mode page

Bit / Byte	7	6	5	4	3	2	1	0
0	PS	SPF (0b)	PAGE CODE (0Ah)					
1	PAGE LENGTH (0Ah)							
2	TST			Reserved		D_SENSE	GLTSD	RLEC
3	QUEUE ALGORITHM MODIFIER				Reserved	QErr		DQue
4	TAS	RAC	UA_INTLCK_CTRL		SWP	Obsolete		
5	APP_TAG_OWN	Reserved				AUTOLOAD MODE		
6	Obsolete							
7								
8	(MSB)	BUSY TIMEOUT PERIOD						
9								(LSB)
10	(MSB)	EXTENDED SELF-TEST COMPLETION TIME						
11								(LSB)

An application tag owner (APP_TAG_OWN) bit set to zero indicates the contents of the DATA BLOCK APPLICATION TAG field may be modified by a device server. An APP_TAG_OWN bit set to one indicates the contents of the DATA BLOCK APPLICATION TAG field shall not be modified by a device server.

If the APP_TAG_OWN bit set to zero the application client shall set the DATA BLOCK APPLICATION TAG field to 0000h and the device server shall ignore the contents of the DATA BLOCK APPLICATION TAG field.

4.1 Vital product data parameters

4.1.1 Vital product data parameters overview and page codes

This subclause describes the vital product data (VPD) page structure and the VPD pages (see table 39) that are applicable to all SCSI devices. These VPD pages are optionally returned by the INQUIRY command (see 6.4) and contain vendor specific product information about a target or logical unit. The vital product data may include vendor identification, product identification, unit serial numbers, device operating definitions, manufacturing data, field replaceable unit information, and other vendor specific information. This standard defines the structure of the vital product data, but not the contents.

Table 39 — Vital product data page codes

Page code	VPD Page Name	Reference	Support Requirements
82h	ASCII Implemented Operating Definition	4.1.2	Optional
01h - 7Fh	ASCII Information	7.6.3	Optional
83h	Device Identification	7.6.4	Mandatory
81h	Obsolete	3.3.7	
84h	Software Interface Identification	7.6.5	Optional
00h	Supported VPD Pages	7.6.6	Mandatory
80h	Unit Serial Number	7.6.7	Optional
xxh	Protection Information	4.1.2	Optional
85h - AFh	Reserved		
B0h - BFh	(See specific device type)		
C0h - FFh	Vendor specific		

4.1.2 Protection Information VPD page (this is a new section)

The Protection Information VPD page (see table 40) provides the application client with the means to obtain certain protection information parameters supported by the logical unit.

Table 40 — Protection Information VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (xxh)							
2	Reserved							
3	PAGE LENGTH (4)							
4	Reserved					GRD_CHK	APP_CHK	REF_CHK
5	Reserved							
6	Reserved							
7	Reserved							

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. If the allocation length is less than the length of the data to be returned, the page length shall not be adjusted to reflect the truncation.

A data block guard check (GRD_CHK) bit set to zero indicates the device server does not check the DATA BLOCK GUARD field before transmitting it to an application client. A GRD_CHK bit set to one indicates the device server does check the DATA BLOCK GUARD field before transmitting it to an application client.

A data block application tag check (APP_CHK) bit set to zero indicates the device server does not check the DATA BLOCK APPLICATION TAG field before transmitting it to an application client. A APP_CHK bit set to one indicates the device server does check the DATA BLOCK APPLICATION TAG field before transmitting it to an application client.

A data block reference check (REF_CHK) bit set to zero indicates the device server does not check the DATA BLOCK REFERENCE TAG field before transmitting it to an application client. A REF_CHK bit set to one indicates the device server does check the DATA BLOCK REFERENCE TAG field before transmitting it to an application client.