To: INCITS Technical Committee T10
From: Kevin Butt
Date: September 12, 2007 1:45 pm
Document: T10/07-374r3 — SSC-3: End-to-end Logical Block Protection

I. Revisions

07-374r0: Initial revision (August 16, 2007) using SSC-3r03d as base.

07-374r1: Incorporate feedback received from Paul Entzel, Dave Peterson, and Gerry Houlder. Due to the large number of changes, change bars are not useful and have been removed.

a. Delete descriptions for FC_CRC and RS_CRC fields.

b. Remove protection of parameter data and CDB - Belongs in SPC-4 or SAM and may be covered by other proposals currently in work in CAP.

c. Added explanation of benefit over protection information applications may already embed in their data.

d. Deleted several sentences and phrases pointed to that added no value or that were likely to not survive letter ballot.

e. Changed a “should” to a “shall” on application client validation and generation of protection information when configured to do so.

f. Some editorial modifications.

g. Added clarifying text about when protection information on data sent in a write is to be validated with respect to the command status.

h. Clarified when protection information on read and on write is required to be supported (i.e., I put it in terms of mode page fields).

i. Removed deferred check condition reporting on Read command when protection information validation fails.

j. Added which sense key to use on reporting validation failure.

k. Deleted the CDBTL field.

l. Added section and figure to describe how protection information is recorded to the medium.

m. Added sections and figures showing the protection information on logical blocks and how the data is laid out for the RECOVER BUFFERED DATA command.

n. Added text to specify the values of the length and endian for each protection method specified.

o. Pointed to updated presentation document 07-373r1.

p. Specified the specific commands that add protection information during transfers.

07-374r2: Incorporate feedback from Bob Nixon and mark the text that could be removed by options currently listed that will likely be removed if nobody argues to keep them. Removed the RDPR and RBDPR fields. Changed the term “end-to-end logical block protection” to “logical block protection”.

07-374r3: Incorporate feedback to 07-374r2 from Paul Entzel, Raymond Gilson, Bob Nixon, Roger Rose, Bill Martin, and Chris Williams.

EDITOR’S NOTE: I believe I have incorporated all comments and suggestions received prior to 12 September 2007 or responded to the reviewer in an email explaining why I am not incorporating them.

II. Introduction

KEY:

~~Deleted Text~~

Added Text

Updates to added text

Potential Removal if nobody fights to keep

EDITOR’S NOTE: <Text>

Questions

Please see SSC-3: Tape end-to-end data protection (Presentation) (07-373r3) as the introduction to this proposal.

IBM Tape has been required to address the question of why our tape drives do not support the T10 standard end-to-end data protection that is available for disk drives. While we have been able to show that it is a disk drive centric solution and that it does not work for tapes, we then have to address the issue of this being one more reason disk should replace tape. That is, disk is viewed as more reliable than tape. We believe that because disk devices have a standard end-to-end data protection they are given an additional marketing tool to win over a tape solution.

IBM believes that SSC-3 needs to provide an end-to-end data protection for tape devices. IBM enterprise drives have been using a proprietary method of end-to-end data protection for over 12 years and we believe that this concept can be easily adapted to fit into a standard.

This end-to-end data protection is accomplished by adding a 32 bit CRC to each data block at the host and transferring that CRC along with the data and validating and storing that CRC with the data on media.

III. Proposal Against SSC-3

EDITOR’S NOTE: All new sections.

EDITOR’S NOTE: Paul Entzel was concerned that I be consistent in my use of “device” versus “device server”, especially in the model clause. I have changed the specific location he pointed to but I still use both terms. Please watch to make sure they are being used correctly and consistently.

4.2.23 End-to-end data protection

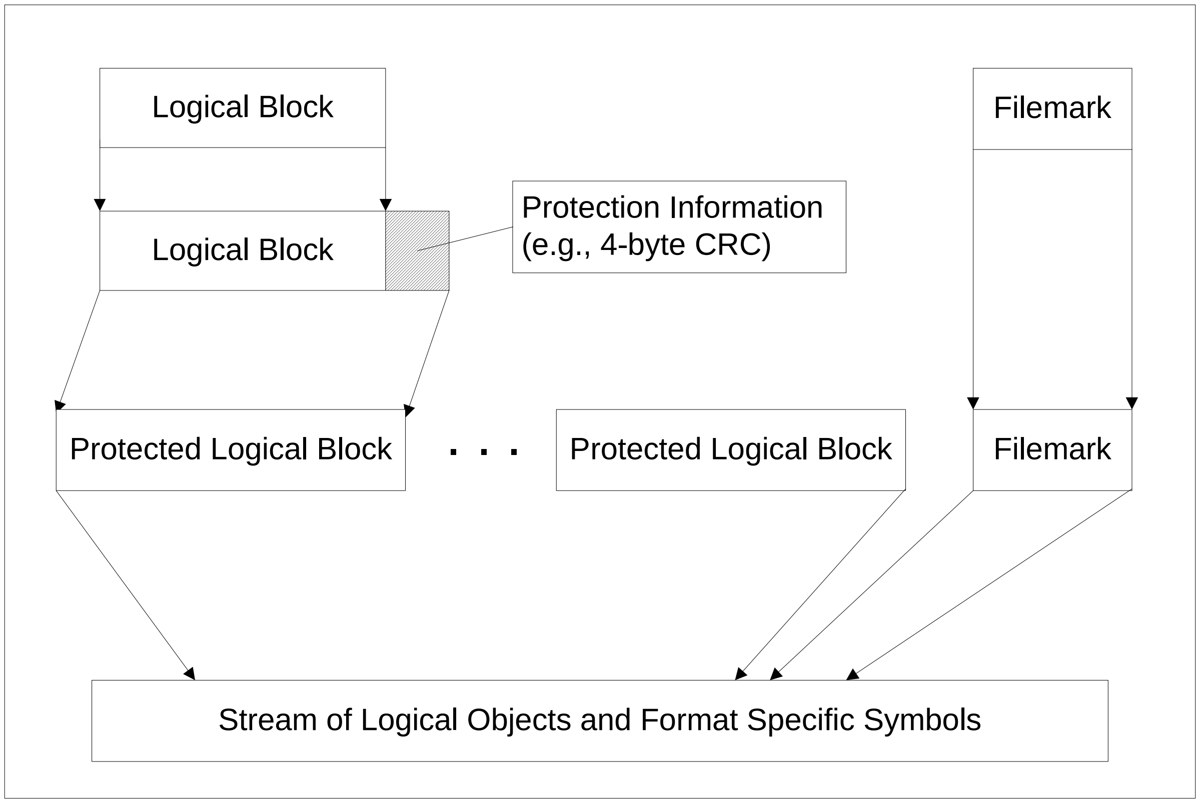
4.2.23.1 End-to-end data protection overview

A device compliant with this standard may contain hardware or software that is capable of checking and/or generating protection information that is transferred with data between the device server and an application client. This protection information transferred with logical blocks is saved to the medium with each logical block and read from the medium with each logical block. This protection information is validated at the destination prior to completing the task thereby ensuring that the data has not been corrupted. This provides a level of detection above what an application client can achieve by inserting its own data protection since the device server validates the protection information. The configuration of this capability is performed using the Control Data Protection mode page (see 8.3.9). A device that supports using this protection information shall support the Control Data Protection mode page.

4.2.23.2 Protection information on a volume

A recorded volume contains logical objects (see 4.2.7.1) and format specific symbols. Logical objects are application client accessible. Format specific symbols are used by the device server to provide methods for recording logical objects on the medium in a manner that allows them to be successfully read at a later date. Often format specific symbols contain information used to protect logical objects. If a device server supports a non-zero value in the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page it should include the protection information field as one of it’s format specific symbols. If the protection information field is not included as one of the device server’s format specific symbols then the protection information field shall be transformed by the device server into a format specific symbol that provides the same level of protection as the protection information field. The format specific symbol that is the protection information field or the transformed protection information field shall be written to the medium with each logical block. The protection information field is accessible by the application client when the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page for the I_T nexus through which that application client communicates is set to a non-zero value. A representation of logical objects and format specific symbols is shown in Figure 1.

FIGURE 1. Protection information shown in relation to logical objects and format specific symbols


Logical Block
Filemark
Logical Block
Protection Information (e.g., 4-byte CRC)
Protected Logical Block
. . .
Protected Logical Block
Filemark
Stream of Logical Objects and Format Specific Symbols

A device server that supports a non-zero value in the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page shall generate the protection information and add it to a logical block before recording the logical block to the medium if the command that transferred the logical block being recorded to medium was received on an I_T nexus for which:

a) the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to zero; or

b) the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to a non-zero value and the WDP bit of the Control Data Protection mode page is set to zero.

A device server that supports a non-zero value in the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page shall read the protection information from the medium, validate it, and remove it from the logical block before transferring the logical block to the application client if the command that is requesting the transfer of a logical block being read was received on an I_T nexus for which:

a) the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to zero; or
b) the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to a non-zero value and the RDP bit of the Control Data Protection mode page is set to zero.

4.2.23.3 Logical blocks and protection information

If the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to zero for a specific I_T nexus then:

a) a READ BLOCK LIMITS command processed through that I_T nexus shall return:
    A) a value in the MINIMUM BLOCK LENGTH LIMIT field greater than or equal to one, and
    B) a value in the MAXIMUM BLOCK LENGTH LIMIT field less than or equal to 2^{24} - 1 bytes;
b) a logical block transferred between the application client and the device server through that I_T nexus is defined by Table x1 if that transfer occurs in response to a:
    A) WRITE(6);
    B) WRITE(16);
    C) VERIFY(6) with the BYTCMP field set to one;
    D) VERIFY(16) with the BYTCMP field set to one;
    E) READ(6);
    F) READ(16);
    G) READ REVERSE(6) with the BYTORD bit set to one; or
    H) READ REVERSE(16) with the BYTORD bit set to one;
    and

TABLE x1. Logical block with no protection information

Byte	Bit							
	7	6	5	4	3	2	1	0
0	Data							
n-1								
n = the TRANSFER LENGTH field specified in CDB for variable length transfers; the BLOCK LENGTH field specified in the mode parameter header (see SPC-4) for fixed block transfers.								

b) a logical block transferred between the application client and the device server through that I_T nexus is defined by Table x2 if that transfer occurs in response to a:
   A) READ REVERSE(6) with the BYTORD bit set to zero; or
   B) READ REVERSE(16) with the BYTORD bit set to zero;

TABLE x2. Logical block for READ REVERSE with BYTORD bit set to zero with no protection information

Byte	Bit							
	7	6	5	4	3	2	1	0
n-1	Data							
0								
n = the TRANSFER LENGTH field specified in CDB for variable length transfers; the BLOCK LENGTH field specified in the mode parameter header (see SPC-4) for fixed block transfers.								

If the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to a non-zero value for a specific I_T nexus then:

a) a READ BLOCK LIMITS command processed through that I_T nexus shall return:
   A) a value in the MINIMUM BLOCK LENGTH LIMIT field greater than or equal to one, and
   B) a value in the MAXIMUM BLOCK LENGTH LIMIT field less than or equal to 2^{24} - 1 - LOGICAL BLOCK PROTECTION INFORMATION LENGTH bytes;

EDITOR'S NOTE: The Read Block Limits modification protects against write without CRC but reading with CRC and not overflowing max block size.

b) a logical block transferred between the application client and the device server through that I_T nexus is defined by Table x3 if that transfer occurs in response to a:
   A) WRITE(6);
   B) WRITE(16);
   C) VERIFY(6) with the BYTCMP field set to one;
   D) VERIFY(16) with the BYTCMP field set to one;
   E) READ(6);
   F) READ(16);
   G) READ REVERSE(6) with the BYTORD bit set to one; or
   H) READ REVERSE(16) with the BYTORD bit set to one;
   and

TABLE x3. Logical block with protection information

Byte	Bit 7	6	5	4	3	2	1	0
0	Data							
n-x-1								
n-x	Protection Information							
n-1								

n = the TRANSFER LENGTH field specified in the CDB for variable length transfers; the BLOCK LENGTH field specified in the mode parameter header (see SPC-4) for fixed block transfers.
x = the LOGICAL BLOCK PROTECTION INFORMATION LENGTH specified in the Control Data Protection mode page.

b) a logical block transferred between the application client and the device server through that I_T nexus is defined by Table x4 if that transfer occurs in response to a:

A) READ REVERSE(6) with the BYTORD bit set to zero; or

B) READ REVERSE(16) with the BYTORD bit set to zero;

TABLE x4. Logical block for READ REVERSE with BYTORD bit set to zero with protection information

Byte	Bit 7	6	5	4	3	2	1	0
0	Data							
n-x-1								
n-x	Protection Information							
n-1								
n = the TRANSFER LENGTH field specified in the CDB for variable length transfers; the BLOCK LENGTH field specified in the mode parameter header (see SPC-4) for fixed block transfers. x = the LOGICAL BLOCK PROTECTION INFORMATION LENGTH specified in the Control Data Protection mode page.								

4.2.23.4 Protection information for Recover Buffered Data

In response to a RECOVER BUFFERED DATA command the device server transfers unwritten data from the logical unit's object buffer to the application client. If the ROBO bit in the Device Configuration mode page (see 8.3.3) is set to zero, then the data is transfered in FIFO (first-in-first-out) order which is the same order in which they were written to the object buffer (see Table x5).

TABLE x5. Data transferred from the logical units object buffer in response to RECOVER BUFFERED DATA command

Byte	Bit 7	6	5	4	3	2	1	0
0	Recover Buffered Data descriptor (first)							
n-1								
n	Recover Buffered Data descriptor (second)							
s								
	.							
	. .							
y	Recover Buffered Data descriptor (last)							
x-1								
n = the block length of the first descriptor x = the number of bytes transferred by a single RECOVER BUFFERED DATA command								

If the ROBO bit in the Device Configuration mode page (see 8.3.3) is set to one, then the data is transfered in LIFO order (last-in-first-out) which is the opposite order from which they were written to the object buffer (see Table x5).

TABLE x6. Data transferred from the logical units object buffer in response to RECOVER BUFFERED DATA command

Byte	Bit							
	7	6	5	4	3	2	1	0
0	Recover Buffered Data descriptor (last)							
n-1								
	. . .							
s	Recover Buffered Data descriptor (second)							
r-1								
r	Recover Buffered Data descriptor (first)							
x-1								

n = the block length of the last descriptor
x = the number of bytes transferred by a single RECOVER BUFFERED DATA command

If the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to zero for a specific I_T nexus then the Recover Buffered Data descriptor used on that I_T nexus is defined by Table x1.

If the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page is set to a non-zero value for a specific I_T nexus then the Recover Buffered Data descriptor used on that I_T nexus is defined by Table x3

4.2.23.5 Protecting logical blocks transferred during writes

When the WDP bit of the Control Data Protection mode page (see 8.3.9) is set to one for a specific I_T nexus, each logical block transferred from the application client through that I_T nexus contains protection information. The commands for which this applies are:

a) WRITE(6);
b) WRITE(16);
c) VERIFY(6) with the BYTCMP field set to one; and
d) VERIFY(16) with the BYTCMP field set to one.

For the WRITE(6) and WRITE(16) commands, the device server shall validate the protection information before the logical block is written to medium. If the FIXED bit in the CDB is set to one each logical block shall be validated before being written to the medium. If the validation of the protection information for a logical block fails, the processing of the command shall terminate prior to writing the failed logical block to the medium. If the validation of the protection information fails, the device server shall respond as defined by the WDPR field in the Control Data Protection mode page. When the WDPR field of the Control Data Protection mode page is set to 00b, the protection information shall be validated before sending status to the command that caused the transfer of the logical block.

For the VERIFY(6) and VERIFY(16) commands with the BYTCMP field set to one the protection information is validated prior to the byte-by-byte compare of the data on the medium and the data transferred from the application client. If the FIXED bit in the CDB is set to one each logical block shall be validated prior to the byte-by-byte compare of the data on the medium for that logical block and the data transferred from the application client for that logical block. If the validation of the protection information fails, the device server shall respond as defined by the WDPR field in the Control Data Protection mode page. If the validation of the protection information does not fail for a logical block, the byte-by-byte compare of the data on the medium for that logical block and the data transferred from the application client for that logical block takes place. This byte-by-byte compare also includes the protection information on the medium and the protection information transferred from the application client.

A device server that supports the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page set to a non-zero value shall support the WDP bit of the Control Data Protection mode page set to one.

An application client that has set the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page to a non-zero value and the WDP bit of the Control Data Protection mode page to one shall add the protection information on each logical block before transferring that data and shall increase the TRANSFER LENGTH field by the length of the logical block protection information. The application client should add the protection information to the data at the earliest point possible. If the data has had the protection information added to the logical block at some point in the application client prior to the hardware that transfers the logical block, the protection information should be validated when it is transferred. If the validation fails, the application client should abort the command and report a status to the user that validation failed.

QUESTION: What is appropriate to do here regarding how the protection information is generated in the application client and it is used? Can I specify this and is this the correct way to specify it?

NOTE yy The device server treats the LOGICAL BLOCK PROTECTION INFORMATION field as the protection information. If the protection information is not added to the logical block, then the validation fails when the bytes used do not validate (e.g., the last 4-bytes of data are treated as the CRC and the last 4-bytes of the data do not calculate as the CRC of the previous data)

4.2.23.6 Protecting logical blocks transferred during reads

If the RDP bit of the Control Data Protection mode page (see 8.3.9) is set to one for a specific I_T nexus, the protection information shall be read from the medium and transferred with the logical block to the application client on that I_T nexus. The commands for which this applies are:

a) READ(6);

b) READ(16);

c) READ REVERSE(6); and

d) READ REVERSE(16).

EDITOR'S NOTE: Bob Nixon asked if we need to cover VERIFY commands with the BYTCMP field set to zero. My thinking is that there is no data begin transferred from the drive or to the drive so it is not needed. However, if the device supports end-to-end then there will be the CRC that could be validated - but the transfer length specified would need to be 4 bytes longer. The verify command may already cover this without additional text. It says:
"A byte compare (BYTCMP) bit of zero specifies the verification shall be simply a medium verification (e.g., CRC, ECC). No data shall be transferred from the application client to the device server."

The protection information shall be validated by the device server before sending status to the command that caused the transfer of the logical block. If the FIXED bit in the CDB is set to one each logical block shall be validated before being transferred to the application client. If the validation of the protection information for a logical block fails, the processing of the command shall terminate prior to transferring the failed logical block to the application client. If the validation of the protection information fails, the device server shall report a Check Condition using a Sense Code of Current Sense, the sense key set to HARDWARE ERROR and the additional sense code set to LOGICAL BLOCK PROTECTION ERROR ON READ. A device that supports the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page set to a non-zero value shall support the RDP bit of the Control Data Protection mode page set to one.

EDITOR'S NOTE: LOGICAL BLOCK PROTECTION ERROR ON READ is a new additional sense code

An application client that has set the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page to a non-zero value and the RDP bit of the Control Data Protection mode page to one should validate the protection information on each logical block at the latest point possible before using the data.

4.2.23.7 Protecting data transferred from the object buffer in response to a RECOVER BUFFERED DATA command

If the RBDP bit of the Control Data Protection mode page (see 8.3.9) is set to one for a specific I_T nexus, each logical block transferred between the device server and the application client on that

I_T nexus during a RECOVER BUFFERED DATA command (see 8.3.9) shall include the protection information. The device server shall:

a) read the protection information from the object buffer if it exists; or
b) generate the protection information if it does not exist.

The protection information for each block shall be validated before sending status to the command. If the validation of the protection information fails for any logical block, the device server shall terminate the command without transferring any additional logical blocks that may exist in the object buffer and report a Check Condition using a Sense Code of Current Sense, the sense key set to HARDWARE ERROR and the additional sense code set to LOGICAL BLOCK PROTECTION ERROR ON RECOVER BUFFERED DATA. A device shall support the RBDP bit of the Control Data Protection mode page set to one if it supports:

a) the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page set to a non-zero value; and
b) the RECOVER BUFFERED DATA command.

EDITOR’S NOTE: LOGICAL BLOCK PROTECTION ERROR ON RECOVER BUFFERED DATA is a new additional sense code

An application client that has set the LOGICAL BLOCK PROTECTION METHOD field of the Control Data Protection mode page to a non-zero value and the RBDP bit of the Control Data Protection mode page to one should validate the protection information on each logical block at the latest point possible before using the data.

5.3 READ(16)

.

.

.

See 4.2.23.6 for the effects of protection information on the transfer of data during a READ(16) command.

5.4 READ REVERSE(16)

.

.

See 4.2.23.6 for the effects of protection information on the transfer of data during a READ REVERSE(16) command.

5.5 VERIFY(16)

.

.

See 4.2.23.5 for the effects of protection information on the transfer of data during a VERIFY(16) command.

5.6 WRITE(16)

.

.

See 4.2.23.5 for the effects of protection information on the transfer of data during a WRITE(16) command.

6.4 READ(6)

.

.

See 4.2.23.6 for the effects of protection information on the transfer of data during a READ(6) command.

6.5 READ REVERSE(6)

.

.

See 4.2.23.6 for the effects of protection information on the transfer of data during a READ REVERSE(6) command.

6.6 VERIFY(6)

.

.

See 4.2.23.5 for the effects of protection information on the transfer of data during a VERIFY(6) command.

6.7 WRITE(6)

.

.

See 4.2.23.5 for the effects of protection information on the transfer of data during a WRITE(6) command.

7.5 READ BLOCK LIMITS

.

.

.

See 4.2.23.3 for restrictions placed on the MAXIMUM BLOCK LENGTH LIMIT when the device server supports protection information.

7.7 RECOVER BUFFERED DATA

The RECOVER BUFFERED DATA command (see table 47) is used to recover data that has been transferred to the logical unit's object buffer but has not been successfully written to the medium. It is normally used to recover the buffered data after error or exception conditions make it impossible to write the buffered data to the medium. One or more RECOVER BUFFERED DATA commands may be required to recover all unwritten buffered data.

If the OBR bit in the Device Configuration mode page (see 8.3.3) is set to one this command is supported by the device server. If the OBR bit in the Device Configuration mode page is set to zero then this command is not supported by the device server.

.

.

.

The processing of this command is similar to the READ(6) command except that the data is transferred from the logical unit's object buffer instead of the medium. The order that descriptor(s) (see 4.2.23.4) ~~logical block(s)~~ are transferred is defined by the ROBO ~~RBO~~ bit in the Device Configuration mode page (see 8.3.3). If the ROBO ~~RBO~~ bit is not implemented, logical block(s) are transferred in the same order they would have been transferred to the medium.

EDITOR'S NOTE: the Device Configuration mode page (see 8.3.3) has:

A recover object buffer order (ROBO) bit of one specifies logical blocks shall be returned from the object buffer of the logical unit on a RECOVER BUFFERED DATA command in LIFO order (last-in-first-out) from that they were written to the object buffer. A RBO (sp) bit of zero specifies logical blocks shall be returned in FIFO (first-in-first-out) order.

.

.

.

8.3.9 Control Data Protection mode page

The Control Data Protection mode page provides controls that allow selective use of end-to-end data protection. The mode page policy of this page shall be Per I_T nexus.

TABLE x7. Control Data Protection mode page format

Byte	Bit							
	7	6	5	4	3	2	1	0
0	PS	SPF(1b)	PAGE CODE (0Ah)					
1	SUBPAGE CODE (F0h)							
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	LOGICAL BLOCK PROTECTION METHOD							
5	Reserved		LOGICAL BLOCK PROTECTION INFORMATION LENGTH					
6	WDP	RDP	RBDP	Reserved				
7	WDPR		Reserved					
8	Reserved							
n								

EDITOR'S NOTE: Gerry Houlder suggested the mode page policy for the Control Data Protection mode page be changed from Per I_T nexus to Shared. I rejected this suggestion because there is not much more complexity by using Per I_T nexus and it removes that possibility of application clients changing it on each other. I have included this note in case others wish to argue for this change.

EDITOR'S NOTE: I wish to leave reserved bytes - at least show that there may be some additional bytes in the future - in case anybody ever decides to do anything on a basis larger than logical block (e.g., file) for end-to-end protection. This seems like a potential extension - although a difficult one.

QUESTION: Should we add the option to pad the protection information? In the only defined protection information a CRC is used. Currently the CRC is added on a byte boundary (i.e., immediately with no padding). If we add a pad to align to a 4-byte boundary, how do we indicate in the data stream the size of the pad? This would add great complexity or require the CRC to be zero neutral and for this reason, I do not want to do it.

The LOGICAL BLOCK PROTECTION METHOD is defined in Table x8.

TABLE x8. END-TO-END DATA PROTECTION METHOD values

Value	Description
00h	Do not use logical block protection
01h	Use the Reed-Solomon CRC as defined in ECMA-319 as the logical block protection information.
02h - FFh	Reserved

The LOGICAL BLOCK PROTECTION INFORMATION LENGTH specifies the length in bytes of the logical block protection information. If the LOGICAL BLOCK PROTECTION METHOD is set to 01h, then the LOGICAL BLOCK PROTECTION INFORMATION LENGTH shall be set to 04h.

The write data protected (WDP) bit set to one indicates that the protection information is included with logical blocks transferred during commands specified in 4.2.23.5. The WDP bit set to zero indicates that the protection information is not included with data transferred when writing. If the LOGICAL BLOCK PROTECTION METHOD field is set to zero, the WDP bit shall be set to zero.

The read data protected (RDP) bit set to one indicates that the protection information is included with logical blocks transferred during the commands specified in 4.2.23.6. The RDP bit set to zero indicates that the protection information is not included with data transferred when reading. If the LOGICAL BLOCK PROTECTION METHOD field is set to zero, the RDP bit shall be set to zero.

The recover buffered data protected (RBDP) bit set to one indicates that the protection information is transferred with the data transferred by the RECOVER BUFFERED DATA command. The RBDP bit set to zero indicates that the protection information is not transferred with the data transferred by the RECOVER BUFFERED DATA command. If the LOGICAL BLOCK PROTECTION METHOD field is set to zero, the RBDP bit shall be set to zero.

The write data protection reporting (WDPR) field is defined in Table x9

TABLE x9. WDPR definition

Value	Device server behavior when the validation of the data fails
00b	Report a Check Condition using a Sense Code of Current Sense, the sense key set to HARDWARE ERROR and the additional sense code set to LOGICAL BLOCK PROTECTION ERROR ON WRITE.
01b	Establish a Check Condition for return on the next eligible command with the Sense Code set to Deferred Sense, the sense key set to HARDWARE ERROR and the additional sense code set to LOGICAL BLOCK PROTECTION ERROR ON WRITE.
10b - 11b	Reserved

EDITOR’S NOTE: LOGICAL BLOCK PROTECTION ERROR ON WRITE is a new additional sense code