ENDL
TEXAS

Date: 1 May 2007
To: T10 Technical Committee
From: Ralph O. Weber
Subject: SPC-4: Extended SCSI Commands

Revision History

r1 Incorporated changes requested by January '07 CAP working group.
r2 Totally rewrote based on discussions in March '07 CAP working group, concepts summarized in 07-158r0, and discussions in April '07 CAP/Security working group (minutes in 07-182).

Because the proposal is a near-total rewrite, change bars have not been applied.

History

In IPsec and FC-SP, transport layer security encapsulates frames in a technology called Encapsulated Security Protocol (ESP). The encapsulation adds security related information to frames that may be otherwise unchanged. An underlying assumption of ESP encapsulation is that the encapsulation data does not need to reference the encapsulated data.

Numerous attempts to add additional data to all SCSI commands (e.g., classification and priority) led to attempts to model an ESP-like encapsulation for SCSI CDBs in revisions 0 and 1 of this proposal. These attempts failed at every turn because the people with plans to add additional data wanted knowledge of the 'encapsulated' CDB's contents to facilitate their command-processing designs.

Starting with r2, this proposal describes a method for appending additional bytes of typed information to CDBs that are defined elsewhere in the SCSI command standards, including but not limited to SPC-4.

Proposed SPC-4 Changes

Reference SPC-4 r08. Blue for new text. ~~Red strikeout~~ for removed text. Green for proposer's notes.

{{New Definition}}

3.1.x Extended CDB (XCDB): A CDB that contains another CDB and additional information to support extra processing (e.g., security features). See 4.3.4.

{{New Acronym}}

XCDB Extended CDB (see 3.1.x)

18484 Preston Road, Suite 102 PMB 178, Dallas, TX 75252 214-912-1373 972-596-2775 Fx ENDL_TX@computer.org

4.3 The Command Descriptor Block (CDB)

4.3.1 CDB usage and structure

A command is communicated by sending a command descriptor block (CDB) to the device server. For several commands, the CDB is accompanied by a list of parameters in the Data-Out Buffer. See the specific commands for detailed information.

If a logical unit validates reserved CDB fields and receives a reserved field within the CDB that is not zero, then the logical unit shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

If a logical unit receives a reserved CDB code value in a field other than the OPERATION CODE field, then the logical unit shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The fixed length CDB formats are described in 4.3.2. The variable length CDB formats are described in 4.3.3. The XCDB format is described in 4.3.4. The CDB fields that are common to most commands are described in 4.3.5. The fields shown in 4.3.2 and 4.3.3 and described in 4.3.5 are used consistently by most commands. However, the actual usage of any field (except the OPERATION CODE field and the CONTROL field) is described in the subclause defining that command. If a device server receives a CDB containing an operation code that is invalid or not supported, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID COMMAND OPERATION CODE.

For all commands, if there is an invalid parameter in the CDB, the device server shall terminate the command without altering the medium.

The XCDB format (see 4.3.5) is a CDB in which additional information is appended to another CDB. The requirements in this subclause apply to the XCDB and the CDB contained in the XCDB.

4.3.2 The fixed length CDB formats

…

4.3.3 The variable length CDB formats

…

4.3.4 Extended CDBs

4.3.4.1 XCDB model

With one exception, any SCSI CDB may be extended in an XCDB. An XCDB shall not be extended in another XCDB. Instead, additional XCDB descriptors shall be added to the existing XCDB.

XCDB descriptors may be:

a) Added by application clients and removed by device servers, or
b) Added and removed by entities outside the scope of this standard that transport or process CDBs and XCDBs during their transfer from an application client to a device server.

4.3.4.2 The XCDB format

The first byte of an XCDB shall contain the operation code 7Eh. The CONTROL byte is the CONTROL byte in the CDB field (see table x1).

Table x1 — ESC CDB format

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Eh)							
1	Reserved							
3								
4	CDB							
n								
XCDB descriptors								
k	First XCDB descriptor							
	. . .							
	Last XCDB descriptor							
l								

The CDB field contains the CDB to which XCDB descriptors are being appended.

{{INVALID XCDB is a new additional sense code.}}

Each XCDB descriptor (see table x2) contains fields associated with a specified extension type (see table x3). The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID XCDB, if:

a) An XCDB does not contain at least one XCDB descriptor;
b) More than one XCDB descriptor contains the same extension type;
c) The order of extension types in the XCDB descriptors differs from that shown in table x3; or
d) More than one XCDB descriptor has the same descriptor order listed in table x3.

Table x2 — XCDB descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	EXTENSION TYPE							
1	Extension parameters							
n								

The EXTENSION TYPE field (see table x3) specifies the size and format of the extension parameters that follow in the XCDB descriptor.

Table x3 — EXTENSION TYPE field

Code	Descriptor Order	Description	Extension size (bytes)	Reference
00h	first	CbCS Extension	tbd	tbd
01h	first	SA Security Extension	tbd	tbd
All others	Reserved			

{{The entries in the above table are only examples. The table is to be incorporated with all code values reserved.}}

The extension parameters contain additional information whose processing is associated with the CDB in the CDB field. The number, content, and size of the extension parameters depends on the contents of the EXTENSION TYPE field.

…

4.3.5 Common CDB fields

4.3.5.1 Operation code

The first byte of a SCSI CDB shall contain an operation code identifying the operation being requested by the CDB. Some operation codes provide for modification of their operation based on a service action (see 4.3.4.2). In such cases, the operation code and service action code combine to identify the operation being requested. The location of the SERVICE ACTION field in the CDB varies depending on the operation code value.

The OPERATION CODE (see table 10) of the CDB has a GROUP CODE field and a COMMAND CODE field. The three-bit GROUP CODE field provides for eight groups of command codes. The five-bit COMMAND CODE field provides for thirty-two command codes in each group. A total of 256 possible operation codes exist. Operation codes are defined in this standard and other command standards (see 3.1.17). The group code value (see table 11) shall determine the length of the CDB.

Table 10 — OPERATION CODE **byte**

Bit	7	6	5	4	3	2	1	0
	GROUP CODE			COMMAND CODE				

The value in the GROUP CODE field specifies one of the groups shown in table 11.

Table 11 — Group Code values

Group Code	**Meaning**	**Typical CDB format**
000b	6 byte commands	see table 3 in 4.3.2
001b	10 byte commands	see table 4 in 4.3.2
010b	10 byte commands	see table 4 in 4.3.2
011b	reserved [a]	
100b	16 byte commands	see table 6 and table 7 in 4.3.2
101b	12 byte commands	see table 5 in 4.3.2
110b	vendor specific	
111b	vendor specific	

[a] The format of the commands using the group code 011b and operation code 7Fh is described in 4.3.3. The format of the commands using the group code 011b and operation code 7Eh is described in 4.3.4. With the exception of operation ~~code~~ codes 7Fh and 7Eh, all group code 011b operation codes are reserved.

…

Annex X
(Informative)
Adding and removing XCDB descriptors

X.1 Adding an XCDB descriptor to a non-extended CDB

An XCDB descriptor is added to CDB other than an XCDB as follows:

1) Generate the XCDB format defined in 4.3.4.2;
2) Copy the CDB into the CDB field of the generated XCDB; and
3) Add the XCDB descriptor with the first byte of the XCDB descriptor placed the first byte following the CDB field.

X.2 Adding an XCDB descriptor to an XCDB

An XCDB descriptor is added to an XCDB as follows:

1) Starting with the first XCDB descriptor following the CDB field (see 4.3.4.2), scan each XCDB descriptor already present in the XCDB and stop the scan when the end of the XCDB descriptors already present in the XCDB is reached or an XCDB descriptor is encounter that conforms to one of the following conditions:
   a) The contents of the EXTENSION TYPE field (see 4.3.4.2) in the scanned XCDB are not recognized (i.e., reserved in the version of this standard upon which the scan is based);
   b) The contents of the EXTENSION TYPE field in the scanned XCDB are identical to the extension type to be added;
   c) The extension type in the scanned XCDB has the same descriptor order assignment (see table x3 in 4.3.4.2) as the extension type to be added; or
   d) The extension type in the scanned XCDB has a descriptor order assignment that requires it to appear after the extension type to be added;
2) Do not add the XCDB descriptor if one of the following is true:
   a) The contents of the EXTENSION TYPE field in the scanned XCDB are identical to the extension type to be added; or
   b) The extension type in the scanned XCDB has the same descriptor order assignment as the extension type to be added;
3) Add the XCDB descriptor only if one of the following is true:
   a) The end of the XCDB descriptors already present in the XCDB has been reached by the scan;
   b) The contents of the EXTENSION TYPE field (see 4.3.4.2) in the scanned XCDB are not recognized; or
   c) The extension type in the scanned XCDB has a descriptor order assignment that requires it to appear after the extension type to be added;
4) If the end of the XCDB descriptors already present in the XCDB has been reached, add the new XCDB descriptor immediately after the last byte of the last XCDB descriptor that is already present; and
5) If the new XCDB is to be added before an already present XCDB descriptor, add the new XCDB descriptor as follows:
   1) Insert the first byte of the new XCDB descriptor immediately following the last byte of:
      a) If the scanned XCDB descriptor is the first XCDB descriptor, after the last byte of the CDB field; or
      b) If the scanned XCDB descriptor is not the first XCDB descriptor, after the last byte of the XCDB descriptor that precedes the scanned XCDB descriptor;
   2) Insert all bytes associated with new XCDB descriptor contiguously following the first byte just inserted; and
   3) Copy all XCDB descriptors already present in the XCDB to the bytes that follow the just inserted XCDB descriptor.

X.3 Removing an XCDB descriptor

An XCDB descriptor is removed from an XCDB by copying all bytes of any remaining XCDB descriptors that follow the removed XCDB descriptor so that they overwrite the removed XCDB descriptor. If no XCDB descriptors remain in the XCDB after the removal, the XCDB is replaced by a CDB that is composed of the bytes in the CDB field.