T10/98-188 r2

Date: Oct. 5, 1998
To: T10 Committee
From: Gerry Houlder, Seagate Technology
Subj: Method for defining very long command blocks

Rev. 1: Adds 4 bytes to the first part of the CDB and changes the 2 byte checksum to a 4 byte CRC at the end of the CDB.

Rev. 2: Corrected several places that referred to wrong number of bytes or called out wrong byte identifier. Deleted the CDB CRC field – its function will be absorbed into the action code specific fields. Added definition of decryption error sense bytes.

This is a follow up to an earlier proposal (98-163). The recommendation from that proposal was to create a method of defining long Command Descriptor Blocks (CDBs) to handle the large number of bytes needed for object oriented commands (also called file oriented commands). These commands are intended for use in Network Attached Storage (NAS) applications. This proposal gives a method of defining long CDBs.

1. A Possible NAS Command Set

I want to give some background on what a command set for these operations would look like. I emphasize that a command set has not been entirely worked out and that this approach has not yet been endorsed by the NAS community. It is only one of several approaches vying for attention. I do want to lay some groundwork regarding a method to define commands with long CDBs, however, before detailing a command set.

This is an outline of some possible object oriented commands. The parameters shown under each command would be the “action code specific fields” for that command. The proposed size of each parameter is in parentheses. Each command would need 8 additional bytes for the bytes shown in the long CDB definition table (later in this document). Note that each command shows a filler field. The purpose of this field is described in the encryption considerations section.

Open Object:
- Partition Number (2 bytes)
- Object Number (4 bytes)
- Stream Index (2 bytes)
- Lock Level (2 bytes)
- Buffer Size (2 bytes)
- Filler (? Bytes)
- DATA IN phase: returns about 48 bytes of stuff.

Read Object Data:
- Partition Number (2 bytes)
- Object Token (4 bytes)
- Read Offset (8 bytes)
- Read Length (4 bytes)
- Filler (? Bytes)
- DATA IN phase: per length requested.

Set Object Attributes:

- Partition Number (2 bytes)
- Object Number (4 bytes)
- File System Attributes (4 bytes)
- Creation Date (8 bytes)
- Last Written Date (8 bytes)
- Last Modified Date (8 bytes)
- Last Access Date (8 bytes)
- Filler (? Bytes)
- No data phase needed.

Close Object:

- Partition Number (2 bytes)
- Object Token (4 bytes)
- Filler (? Bytes)
- No data phase needed.

Write Object Data:

- Partition Number (2 bytes)
- Object Token (4 bytes)
- Write Offset (8 bytes)
- Write Length (4 bytes)
- Filler (? Bytes)
- DATA OUT phase: per length requested.

This would not be a complete command set for object oriented commands, but is representative of the lengths needed. The command in this group with the most defined bytes defined (Set Object Attributes) needs 42 bytes + 10 bytes = 52 bytes minimum. The filler field should include space for CRC or other verification bytes so verification of proper decryption is possible. Adding 4 bytes for this would result in a command set with 56 byte length for all commands.

2. Encryption Considerations

There are several schools of thought concerning data security. The consensus is that we cannot depend on the entire network being secure, so some level of encryption is needed to provide data security. The distribution of encryption keys throughout the network requires some secure distribution method outside the scope of the storage network.

One method is to encrypt the data at the application before sending it to the network. The storage device is unaware of whether the data is encrypted or not and simply stores it or retrieves it as requested. This doesn’t require any new features in the storage device so we don’t need to concern ourselves with this.

Another method only encrypts the data but requires the storage device to decrypt it before storing it. The storage device will usually be required to encrypt the data when transferring it to an initiator but will normally use a different encryption key than the one used for the write operation. This requires the storage device to maintain a number of different encryption keys and the command must identify which of the keys should be used. This is why an encryption identification field is needed. There could be a number of “assumed” keys on a per initiator basis, but there may be different users coming from the same initiator that must have different levels of access. The network may also desire to assign a different key to each group of objects and a user must have that key in order to access that group of objects.

A third method, for systems that want even higher security, is to also encrypt the command information in addition to the data information. The encryption identification field can be used to indicate this also. In my long CDB proposal, the first 8 bytes of the CDB are never encrypted. When the encryption identification field (which is within the 8 unencrypted bytes) indicates that the CDB is encrypted, the storage device must decrypt the rest of the CDB bytes. The action code specific fields should include CRC or other verification bytes so the target can verify that the command has been properly decrypted.

When the CDB is encrypted, it is important that all commands be the same size. If they weren’t, then an unauthorized entity could tell what commands were what based on the size. This is where the filler bytes come in. When the command set is designed, each command will have enough filler bytes added so that all commands will be a standard size.

3. Long CDB Definition

Table xxx – Long CDB definition

Bit / Byte	7	6	5	4	3	2	1	0
0	Long Command Op Code (7Fh)							
1	Control byte							
2	Reserved							
3	Reserved							
4	Reserved							
5	Encryption Identification							
6	Reserved							
7	Additional CDB Length (n-7)							
8	(MSB)	Action Code						
9								(LSB)
10								
-		Action code specific fields						
n								

The long command op code value will be chosen from the list of currently reserved op codes. One of the group 3 op codes (0x7F) is my recommendation.

The control byte is the same control byte that is the last byte in the 6, 10, 12, and 16 byte CDB structures.

The encryption identification field indicates whether CDB bytes 8 through n and/or the data bytes are encrypted. The value also indicates which encryption key to use for decryption. A value of zero indicates no encryption.

The additional CDB length field indicates the number of additional CDB bytes. This number shall be a multiple of 4.

The action code field indicates the action being requested by the application client. This is equivalent in handling to the operation code field in all other commands today.

Each action code will define a number of action code specific fields that are needed for that action. These will be described in the clause defining that action code.

If the device server detects an error during decryption of encrypted CDB bytes, it shall return CHECK CONDITION status and set sense bytes to CDB DECRYPTION ERROR (24h/01h). If the device server detects an error during decryption of encrypted data bytes, it shall return CHECK CONDITION status and set sense bytes to DATA DECRYPTION ERROR (26h/05h).