Capability based Command Security

SCSI commands standard proposal

IBM Research Lab in Haifa

February 2007

Table of Contents

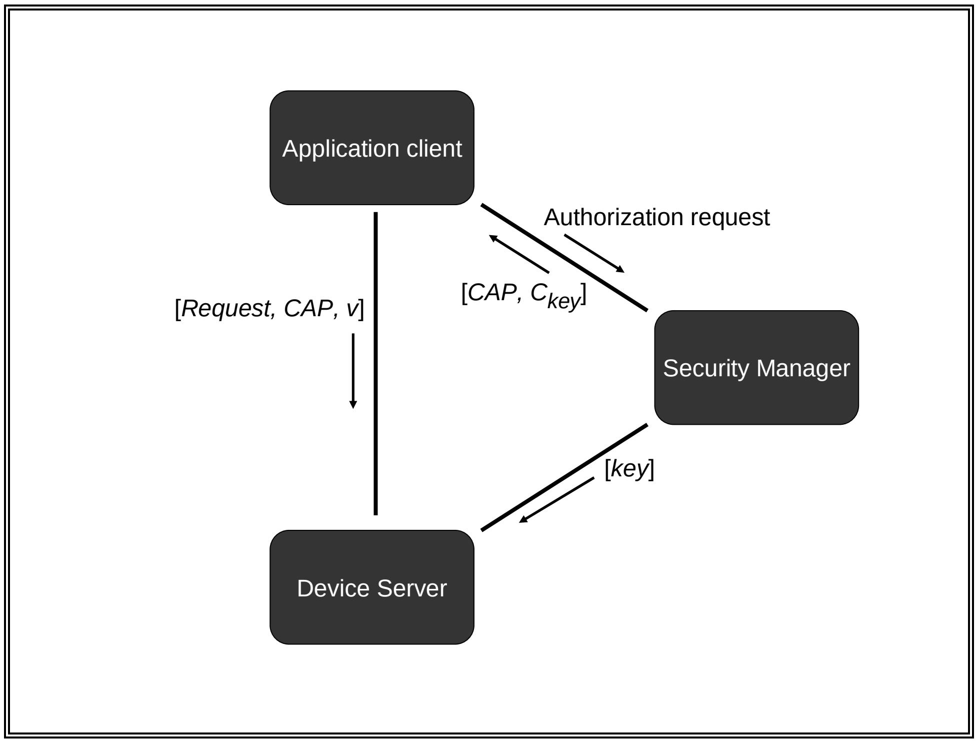
1 General

1.1 Overview

The Capability based Command Security (CbCS) protocol is a capability-based SCSI protocol which cryptographically enforces the integrity of the credential and its legitimate use by the client. It is a proposed extension to the SCSI Primary Commands (SPC) standard. The protocol is based on the OSD security protocol [ObsSec05] [OSD04], mapping the object access control and security protocol to logical units of any device type, with appropriate adjustments.

CbCS is a credential-based access control protocol composed of three active entities: an application client, a device server, and a security manager (see Figure 1). As a capability-based access control system, all requests to the device server must be accompanied by a capability, which encodes a set of rights the holder has on a logical unit, and be cryptographically secured.

Figure 1 - CbCS architecture



There are two steps required by the client to access a logical unit via SCSI commands:

- Prior to sending an I/O command to a SCSI target device, the client requests a credential with certain permissions from the security manager (over a secure channel) and in return the security manager sends back a pair [*CAP*, C_{key}]. Together, the capability *CAP* and capability key C_{key} form the credential. C_{key} is secret whereas *CAP* is public.
- The client sends to the SCSI target device, together with the request, the capability *CAP* along with a validation tag *v*, namely [*Request*, *CAP*, *v*]. *v* is computed by the client using C_{key}.

In order to ensure the authenticity and privacy of the commands and the data, the application client has to communicate with both the security manager and the device server over a secure channel.

As will be described below, C_{key} can be computed from *CAP* by the device server; hence the validation tag is also computable by the device server. Based upon the security method, the device server validates the validation tag and checks whether the operation requested by the command is indeed permissible by the capability *CAP*. Note that the device server does not need to authenticate the client or to have a notion of client identity.

Throughout the protocol there is a need to use a pseudo-random function as a cryptographic primitive. A field in capability specifies which function is used to compute the validation tag for the command.

The security manager and the device server share a set of symmetric secret keys that are updated periodically. A credential is based on one of the keys. When a key is updated, all credentials based on that key are no longer valid.

The details of the protocol are described and explained inside the following section, which is intended to be incorporated in SPC-4.

1.2 The Security Manager

The security manager is responsible for (1) key management (both storing and rotating the keys) across all of the target devices managed by the security manager and (2) composing appropriate credentials for a client according to a given policy.

To enable flexibility in the upper software layers, the precise security manager policies are not defined by this protocol. The standard specifies the format of the credential, which is an entity exchanged between the client, the security manager and the target device; however, the format of the policies and the methods used for determining and generating credentials based on those policies are internal to the security manager and are left out of the scope of the protocol.[1]

Issues such as security method, and the type of communications channel between client and security manager are determined by the upper-layer software. This allows the upper-layer software to make decisions based on environment (e.g., secured SAN vs. insecure LAN) and usage models (e.g., sensitive corporate data vs. open web server data).

A protocol for setting and refreshing the keys and controlling security attributes on the target devices is standardized in by means of SCSI commands. For this purpose the security manager acts as a target device. It is possible to pass those commands and associated data transfers as payload of data inside any type of communications channel between the security manager and the target device, and handle its completion and status via that channel.[2]

Likewise, a protocol for an application client to authenticate and retrieve credentials from the security manager is standardized in by means of SCSI commands. Again, it is possible to pass those commands and associated data transfers as payload of data inside any type of communications channel between the security manager and the target device, and handle its completion and status via that channel.[3]

[1] The security manager may communicate with a separate policy manager for access decision making.
[2] Using non-SCSI mechanism to exchange the keys may be useful to save having the SCSI stack and/or FC connectivity in the security manager.
[3] Using non-SCSI mechanism to exchange the credentials may be useful to save having the SCSI stack and/or FC connectivity in the security manager.

To ensure correctness of the security protocol itself, the interactions between the security manager and the target device are defined using the capability model. Device servers require the security manager to present a valid capability authorizing the operation. Because security-level commands such as changing keys require a high degree of security, the security manager must use the appropriate method of security as specified for the device server with which it is interacting. This prevents the potential security weakness of attempting to set a key using a command with the NOSEC security method.

Finally, the capability includes an expiration time that limits the time a compromised capability can cause damage and helps upper-level distributed software by limiting the time a client can access a logical unit. Because the security manager encodes the expiration time within the capability and the device server interprets the expiration time, we require some degree of clock synchronization between the device server and Security manager. The protocol for synchronizing the clocks is not specified as part of the protocol, but expects that a standard clock synchronization protocol is implemented in a secure manner. Unauthorized setting of the target device's clock could cause denial of service or re-enabling of previously revoked access.

1.3 Special Considerations

Credential Revocation: Rapid revocation of access rights is a very important operation for distributed systems. Reasons for revocation could be security breach detection and SAN configuration changes. The protocol recognizes this basic function and enables immediate revocation via two mechanisms for invalidating a credential.

The first mechanism, key exchange, is a coarse-grained approach that exchanges the key between the security manager and the device server. By changing the shared key, all previous credentials a security manager had generated for the logical units of a particular device are now invalid.

The second mechanism is fine-grained and invalidates all outstanding credentials for a given logical unit by utilizing the LU policy access tag. This tag is a settable LU attribute (typically by the security manager only, since it requires a special permission to set). A valid credential must match the policy access tag of the logical unit; hence, we can invalidate all outstanding credentials for a logical unit by modifying the value of its policy access tag. In order to avoid re-validation of revoked credential, policy access tags should not be reused within the lifetime of a credential.

Bootstrapping: Since a credential includes the policy access tag, which is stored as an attribute for the logical unit, we may have a problem of bootstrapping, particularly if the security manager does not have this information in its memory. How does the security manager generate a credential to read this attribute if it does not know this attribute?

To address this, the security manager has the option to generate a capability with a wild card (zero) for the policy access tag. In this case, when calculating the capability arguments, the device server should not take into account the actual value of the policy access tag associated with the logical unit. This essentially creates a credential which cannot be invalidated during its life-span other than by a key exchange.

Delegation: To delegate a credential to another client, a client must transfer both *CAP* and C_{key} in a private manner (e.g., over an encrypted channel). If desired, such delegation may be prevented by use of the `Audit` field. As shown in [HKN05], an appropriate use of the `Audit` tag leads to "security confinement" thereby preventing leakage of information to unauthorized users.

Migration: The protocol has to take in account gradual migration of SANs to using access controls to logical units. First, host support and storage systems support are likely to be available at different times from different vendors. Security attributes should be settable and retrievable at the LU level. Specifically:

- A target port should be able to support both regular and secure LUs at the same time.
- An application client should be able to determine the security access controls applied to any particular LU.
- Compatibility should be maintained for old application clients such that rejecting their access to secure LUs is done in a way that allows for graceful failures. Ideally the secure LUs would not look like usable LUs to back-level application clients. Repeated errors are too much of an impact for host systems.

2 Changes to SPC-4

Proposal 07-029 defines ESC (Encapsulated SCSI Command) CDB format. This proposal is dependent on 07-029r1.

2.1 ENCAPSULATION TYPE definitions

Following is a proposed change to table x3 from 07-029r1.

Table x3 — OUTERMOST ENCAPSULATION TYPE field and NEXT ENCAPSULATION TYPE field

Code	Description	Size of encapsulation parameter descriptors (bytes)		Reference
		Prefix	Postfix	
00h	No next layer	n/a	n/a	4.3.4.2.1
Prefix and Postfix codes				
01h – 07h	Reserved			
Prefix only codes				
xxh	Capability based Commands Security	98	0	
xxh - FFh	Reserved		0	

2.2 PERFORM SECURE command

New sub-section in section 6:
6.X PERFORM SECURE command

The PERFORM SECURE command (see Table 1) allows the application of security to a SCSI command using the parameters specified in this subclause.

The PERFORM SECURE command is mandatory if the CbCS bit in standard INQUIRY data (see 2.4) is set to one.

The encapsulated CDB may be any valid CDB (i.e., any CDB defined in any SCSI standard).

Table 1 - PERFORM SECURE COMMAND CDB format

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Eh)							
1	DATA TRANSFER		Reserved					
2	Reserved							
3	OUTERMOST ENCAPSULATION TYPE (10h)							

Bit Byte	7	6	5	4	3	2	1	0
4	NEXT ENCAPSULATION TYPE (0)							
5	Reserved							
6 63	Capability descriptor							
64 103	Security Parameters							
104 n	Encapsulated CDB							

The DATA TRANSFER field is defined in 4.3.4.2.1. (Ed note: see 07-029)

The Capability field is defined in **Error! Reference source not found.**.

The Security Parameters field is defined in **Error! Reference source not found.**.

The Encapsulated CDB field contains the SCSI CDB.

The contents of the Capability descriptor (see table 3) enables the device server to verify that the requesting application client is allowed to perform the command specified in the encapsulated CDB.

Table 2 – Capability descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	CAPABILITY FORMAT (1h)				KEY VERSION			
1	Reserved							
2	(MSB)	SECURITY METHOD						
3								(LSB)
4	(MSB)	INTEGRITY CHECK VALUE ALGORITHM						
5								(LSB)
6	(MSB)	CAPABILITY EXPIRATION TIME						
11								(LSB)
12	AUDIT							
31								
32	PERMISSIONS BIT MASK descriptor							
35								
36	(MSB)	POLICY ACCESS TAG						
39								(LSB)
40	Reserved				LU DESCRIPTOR TYPE			
41	LU DESCRIPTOR LENGTH							
42	LU DESCRIPTOR							
57								

The CAPABILITY FORMAT field (see Table 3) specifies the format of the capability. The capability format may also be the credential format. This standard only supports the CAPABILITY FORMAT field being set to 1h (i.e., the format defined by this standard). The device server shall verify that the command functions requested in the PERMISSIONS BIT MASK field (see table 5) are permitted (see 2.3).

Table 3 – Capability format field

Code	Description
0h	No capability
1h	The format defined by this standard
2h - Fh	Reserved

The command shall be terminated with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB, if the CAPABILITY FORMAT field contains any value other than 1h.

The KEY VERSION field is specifies which secret key, from the set of working keys, is being used to compute the integrity check value.

The SECURITY METHOD field should be set a value defined in Table 4. Security methods are specified in 2.5.2.

Table 4 specifies the security methods. Those values shall be used to specify security methods to be used by this command.

Table 4 - Security method field

Code	**Security Method**	**Description**
0000h	NOSEC	No security (see 2.5.2.3)
0001h	CAPKEY	Integrity of capabilities (see 2.5.2.4)
0002h – 0FFFh	Reserved	
1000h – FFFEh	Vendor specific	
FFFFh	Reserved	

A command prepared for a security mode other than the one specified in the CDB SECURITY METHOD field may complete without errors (e.g., a command prepared for the CAPKEY security method may complete without errors reported by the device server if the NOSEC security method is in use).

If the value of the SECURITY METHOD field is CAPKEY, the integrity of the CDB shall be validated (see 2.5.2.4) before any other command processing actions are undertaken (i.e., before verifying that the command function requested in the CDB is permitted by the capability).

(Ed Note: The above two paragraphs look like they belong in 2.5.2 not here)

The INTEGRITY CHECK VALUE ALGORITHM field specifies the algorithm used to compute the integrity check value for this capability (Table 10).

The CAPABILITY EXPIRATION TIME field specifies expiration time of the capability. The time is the number of milliseconds that have elapsed since midnight, 1 January 1970 UT. If the CAPABILITY EXPIRATION TIME field is non-zero and is less than the current time set in the device server when processing the command, the command shall be terminated with a CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN CDB.

If the CAPABILITY EXPIRATION TIME field contains zero, the capability has no expiration time.

The method for synchronizing the clocks is outside the scope of this standard.

The AUDIT field contains a vendor specific value

The PERMISSIONS BIT MASK descriptor (see Table 5) specifies the permissions allowed by this capability. More than one permissions bit may be set. The device server shall verify that the bits applicable to the encapsulated command are all set to one in the PERMISSIONS BIT MASK descriptor before performing the encapsulated SCSI command. In table 5 byte 0 and byte 1 specify the permissions for all SCSI commands. In table 5 byte 2 and byte 3 may be used by a device type specific command set standard to specify permissions unique to the device type. However, other command set standards shall not override the definition of table 5 byte 0 and byte 1 as defined in this standard. The association of the permissions specified in the permissions but mask descriptor and SCSI commands is specified in 2.3.

Table 5 - Permissions bit mask descriptor

Bit / **Byte**	7	6	5	4	3	2	1	0
0	DATA READ	DATA WRITE	ATTR READ	ATTR WRITE	SEC MGMT	Reserved		
1	Reserved							
2	For device type specific use							
3								

A DATA READ bit set to zero indicates the encapsulated SCSI command has no read permission. A DATA READ bit set to one indicates the encapsulated SCSI command has read permission.

A DATA WRITE bit set to zero indicates the encapsulated SCSI command has no write permission. A DATA WRITE bit set to one indicates the encapsulated SCSI command has write permission.

A attribute read (ATTR READ) bit set to zero indicates the encapsulated SCSI command has no attribute read permission. A ATTR READ bit set to one indicates the encapsulated SCSI command has attribute read permission.

A attribute write (ATTR WRITE) bit set to zero indicates the encapsulated SCSI command has no attribute write permission. A ATTR WRITE bit set to one indicates the encapsulated SCSI command has attribute write permission.

A security management (SEC MGMT) bit set to zero indicates the encapsulated SCSI command has no security management permission. A ATTR WRITE bit set to one indicates the encapsulated SCSI command has security management permission.

The LU DESCRIPTOR TYPE field (see Table 6) specifies the format of information in the LU DESCRIPTOR field.

Table 6 - LU DESCRIPTOR TYPE field

Code	**Description**
0h	Vendor specific
1h – 2h	Reserved
3h	NAA (see 7.6.3)

Code	Description
4h – Fh	Reserved

In order to implement credential based on logical unit unique identifier, the same identifier type shall be 1) used in the access control policies set in the policy manager; 2) returned by the device server in Device Identification VPD page (Inquiry page 83h); 3) used by the application client to identify the device and request the corresponding credential from the security manager.

(Ed Note: The above paragraph look like it belongs in 2.5.2 not here)

The LU DESCRIPTOR LENGTH field indicates the length in bytes of the LU DESCRIPTOR field. If the value of this field is greater than 16, the command shall be terminated with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The security parameters field (see Table 7) contains the security information for the PERFORM SECURE COMMAND command.

Table 7 - Security parameters field

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	REQUEST INTEGRITY CHECK VALUE						
19								(LSB)

TODO: Have to extend the size – make it variable length to accommodate new integrity check value algorithms that produce new sizes.

```
Note: The reason we "grouped together" a single field, is that
the security parameters are likely to be extended in the future
to include other fields in addition to the integrity check value.
```

The REQUEST INTEGRITY CHECK VALUE field contains an integrity check value for the request sent by the application client. The REQUEST INTEGRITY CHECK VALUE field shall only be used by the CAPKEY security method (see 2.5.2.4) and shall be used to validate the Capability.

The device server shall validate the request integrity check value as described in 2.5.2.4.

2.3 Association between commands and command functions

The PERMISSIONS BIT MASK descriptor (see Table 5) specifies which command functions are allowed by this capability. When processing PERFORM SECURE commands, the device server shall verify that the bits applicable to the encapsulated SCSI command are all set to one in the PERMISSIONS BIT MASK descriptor before performing the request specified by the encapsulated SCSI command.

The associations of permissions as specified in the PERMISSIONS BIT MASK descriptor is not specified by this standard. Other command set standards may specify required associations between commands and permissions pertaining to the specific device type.

The associations between commands and permissions in the device server may be queried by an application client using the REPORT SUPPORTED OPERATION CODES command (see 6.24).

Following are proposed changes to the REPORT SUPPORTED OPERATION CODES command.

Change in section 6.24 REPORT SUPPORTED OPERATION CODES command

6.24.2 All_commands parameter data format

Each command descriptor (see table 169) contains information about a single supported command CDB.

Table 169 — Command descriptor format

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	Reserved							
2	(MSB)	SERVICE ACTION						
3								(LSB)
4	Reserved							
5	Reserved					PBMP	CTDP	SERVACTV
6	(MSB)	CDB LENGTH						
7								(MSB)
8 - 19	Command timeouts descriptor, if any (see 6.24.4)							
20 - 23	Permissions bit mask, if any							

The OPERATION CODE field contains the operation code of a command supported by the logical unit.

The SERVICE ACTION field contains a supported service action of the supported operation code indicated by the OPERATION CODE field. If the operation code indicated in the OPERATION CODE field does not have a service action~~s~~, the SERVICE ACTION field shall be set to 00h.

A permissions bit mask present (PBMP) bit set to one indicates that the permissions bit mask for the command (see Table 5) is included in this command descriptor. A PBMP bit set to zero indicates that the permissions bit mask is not included in this command descriptor.

A command timeouts descriptor present (CTDP) bit set to one indicates that the command timeouts descriptor (see 6.24.4) is included in this command descriptor. A CTDP bit set to zero indicates that the command timeouts descriptor is not included in this command descriptor.

A service action valid (SERVACTV) bit set to zero indicates the operation code indicated by the OPERATION CODE field does not have service actions and the SERVICE ACTION field contents are reserved. A SERVACTV bit set to one indicates the operation code indicated by the OPERATION CODE field has service actions and the contents of the SERVICE ACTION field are valid.

The CDB LENGTH field contains the length of the command CDB in bytes for the operation code indicated in the OPERATION CODE field, and if the SERVACTV bit is set to the service action indicated by the SERVICE ACTION field.

If the RCTD bit is set to one in the REPORT SUPPORTED OPERATION CODES CDB (see 6.24.1), the command timeouts descriptor (see table 172 in 6.24.4) shall be included. If the RCTD bit is set to zero, the command timeouts descriptor shall not be included.

If the PBMP bit is set to one in the REPORT SUPPORTED OPERATION CODES CDB (see 6.24.1), the permissions bit mask for the command (see Table 5) shall be included. If the PBMP bit is set to zero, the command timeouts descriptor shall not be included.

If the permissions bit mask field is included the device server shall only process theis command if:

a) the command is encapsulated in a PERFORM SECURE command (see 2.2);

b) the capabilities allow processing; and

c) the permissions match those specified by the PERMISSIONS BIT MASK descriptor of the Capability descriptor (see **Error! Reference source not found.**)

If the RCTD bit is set to zero and the PBMP bit is set to one, the command timeouts descriptor shall be set to zero.

6.24.3 One_command parameter data format

The REPORT SUPPORTED OPERATION CODES one_command parameter data format (see table 170) contains information about the CDB and a usage map for bits in the CDB for the command specified by the REPORTING OPTIONS, REQUESTED OPERATION CODE, and REQUESTED SERVICE ACTION fields in the REPORT SUPPORTED OPERATION CODES CDB.

Table 170 — One_command parameter data

Byte \ Bit	7	6	5	4	3	2	1	0
0	Reserved							
1	CTDP	PBMP	NUSAGES			SUPPORT		
2	(MSB) CDB SIZE (n-3)							
3								(LSB)
4	1st CDB USAGE DATA							

Byte \ Bit	7	6	5	4	3	2	1	0
n								
n+1 … n+12	1st Command timeouts descriptor, if any (see 6.24.4)							
n+13 … n+16	1st Permissions bit mask, if any							
	.....							
m+1 … m+n-3	Last CDB USAGE DATA							
m+n-2 … m+n+9	Last Command timeouts descriptor, if any (see 6.24.4)							
m+n+10 … m+n+13	Last Permissions bit mask, if any							

A command timeouts descriptor present (CTDP) bit set to one indicates that the command timeouts descriptor (see 6.24.4) is included in the parameter data. A CTDP bit set to zero indicates that the command timeouts descriptor is not included in the parameter data.

A permissions bit mask present (PBMP) bit set to one indicates that the permissions bit mask for the command (see Table 5) is included in this command descriptor. A PBMP bit set to zero indicates that the permissions bit mask is not included in this command descriptor.

The NUSAGES field specifies the number of CDB USAGE DATA items that are present in addition to the first one. A value of zero specifies that only a single item is present.

Since the overall supported usage data for the command is the bitwise OR of all the present CDB USAGE DATA fields, this OR'ed bitmap must be specified as one of the present items, and it must be the first item in the list. In other words, the value of 1st CDB USAGE DATA must be equal to the bitwise OR of all the CDB USAGE DATA items. Or in other words, any bit set to one in any of the CDB USAGE DATA items must be set to one in the 1st CDB USAGE DATA item.

(Ed Note: The idea of have more that one CDB returned per the one-command parameter list is not a good idea and should be removed. Multiple CDBs should be returned in multiple one-command parameter lists. I also question using this command to return any permissions information. If that information is required it should be defined ina new command.)

The SUPPORT field is defined in table 171.

Table 171 — SUPPORT values

Support	Description
000b	Data about the requested SCSI command is not currently available. All data after byte 1 is not valid. A subsequent request for command support data may be successful.
001b	The device server does not support the requested command. All data after byte 1 is undefined.
010b	Reserved
011b	The device server supports the requested command in conformance with a SCSI standard. The parameter data format conforms to the definition in table 170.
100b	Reserved
101b	The device server supports the requested command in a vendor specific manner. The parameter data format conforms to the definition in table 170.
110b – 111b	Reserved

The CDB SIZE field contains the size of the CDB USAGE DATA field in the parameter data, and the number of bytes in the CDB for command being queried (i.e., the command specified by the REPORTING OPTIONS, REQUESTED OPERATION CODE, and REQUESTED SERVICE ACTION fields in the REPORT SUPPORTED OPERATION CODES CDB).

The CDB USAGE DATA field contains information about the CDB for the command being queried. The first byte of the CDB USAGE DATA field shall contain the operation code for the command being queried. If the command being queried contains a service action, then that service action code shall be placed in the CDB USAGE DATA field in the same location as the SERVICE ACTION field of the command CDB. All other bytes of the CDB USAGE DATA field shall contain a usage map for bits in the CDB for the command being queried.

The bits in the usage map shall have a one-for-one correspondence to the CDB for the command being queried. If the device server evaluates a bit in the CDB for the command being queried, the usage map shall contain a one in the corresponding bit position. If any bit representing part of a field is returned as one, all bits for the field shall be returned as one. If the device server ignores or treats as reserved a bit in the CDB for the command being queried, the usage map shall contain a zero in the corresponding bit position. The usage map bits for a given CDB field all shall have the same value.

For example, the CDB usage bit map for the REPORT SUPPORTED OPERATION CODES command is: A3h, 0Ch, 03h, FFh, FFh, FFh, FFh, FFh, FFh, FFh, 00h, 07h. This example assumes that the logical unit only supports the low-order three bits of the CDB CONTROL byte. The first byte contains the operation code, and the second byte contains three reserved bits and the service action. The remaining bytes contain the usage map.

If the RCTD bit is set to one in the REPORT SUPPORTED OPERATION CODES CDB (see 6.24.1), the command timeouts descriptor (see table 172 in 6.24.4) shall be included. If the RCTD bit is set to zero, the command timeouts descriptor shall not be included.

If the PBMP bit is set to one in the REPORT SUPPORTED OPERATION CODES CDB (see 6.24.1), the permissions bit mask (see Table 5) shall be included. If the PBMP bit is set to zero, the permissions bit mask shall not be included.

If the RCTD bit is set to zero and the PBMP bit is set to one, the command timeouts descriptor bytes for each CDB USAGE DATA item shall be set to zero.

6.24.4 Command timeouts descriptor

<No changes in this section>

6.24.5 Permissions Bit mask

The Permissions Bit Mask specifies the permissions for which authorization is required when the command is used as an encapsulated command in the PERFORM SECURE COMMAND command (see 2.2). The definition of this field is as defined in 2.3.

For each device type, the device specific standard command set or this standard may require certain authorization types (command functions) to be associated with certain commands. The returned Permissions Bit Mask should conform to these requirements.

```
Should we provide a command for setting permissions bit masks for
a command?

Should we specify permissions bit masks for SPC-4 commands in
SPC-4?
```

2.4 Standard INQUIRY data

Change in 6.4.2 Standard INQUIRY data:

The standard INQUIRY data (see table 85) shall contain at least 36 bytes.

Table 85 — Standard INQUIRY data format

Bit / Byte	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	TPGS		3PC	**CbCS**	Reserved	PROTECT
6	Obsolete	ENCSERV	VS	MULTIP	Obsolete	Obsolete	Obsolete	ADDR16[a]
7	Obsolete	Obsolete	WBUS16[a]	SYNC[a]	Obsolete	Obsolete	CMDQUE	VS
8	(MSB)	T10 VENDOR IDENTIFICATION						
15								(LSB)
16	(MSB)	PRODUCT IDENTIFICATION						
31								(LSB)

Byte \ Bit	7	6	5	4	3	2	1	0
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 transport protocols other than the SCSI Parallel Interface, these fields are reserved.

A Capability based Command Security (CbCS) bit set to one indicates that the SCSI target device supports Capability based Command Security (see 2.2). A CbCS bit set to zero indicates that the SCSI target device does not support Capability based Command Security.

If the CbCS bit is set to one, the target device is required to support the following commands:

- PERFORM SECURE command (see 2.2);
- REPORT SUPPORTED OPERATION CODES (see 2.3); (Editors comment - I don't agree with this method)
- SET KEY command (see 2.6); and
- SET MASTER KEY command (see 2.6.3).

2.5 Security attributes

2.5.1 Overview

Device servers supporting CbCS may support more than one security attribute. Security attributes may be:

a) target device based;

b) logical unit based;

c) changeable; or

d) non-changable.

Device servers may support:

a) only target device based security attributes;

b) only logical unit based security attributes; or

c) both.

If a device server supports both and a logical unit based security attribute is received, then the logical based security attribute overrides any target based security attribute on that device server.

Global attributes are queried and modified through the SECURITY PROTOCOL well-known logical unit (see x.x.x)

Table 8 specifies the security attributes.

Table 8 - Security attributes

Security attribute name	**Length (bytes)**	**Target device or logial unit - specific**	**Application client settable**
Supported security methods	n*2	Target device	No
Default security method	2	Target device	Yes
Security method	2	Target device/Logical unit	Yes
Supported integrity check value algorithms	n*2	Target device	No
Supported DH groups	n*2	Target device	No
Clock	6	Target device	No
key identifier	7	Target device / Logical unit	No
LU initial policy access tag	4	Target device	Yes
Policy access tag	4	Logical unit	Yes
Security token	Not defined [c]	Logical unit [b]	No

[b] The security token returned by the device server is unique to the I_T_L nexus on which the security token is returned.

[c] The security token length is specific to the implementation of secure channel for the I_T_L nexus

The supported integrity check value algorithms security attribute provides a means for a device server to report the integrity check value algorithms it supports. See 2.5.3.

The supported DH group security attribute provides a means for a device server to report the DH groups it supports for the KEY EXCHANGE phase of the SET MASTER KEY command (2.6.3). See 2.5.4.

The clock security attribute shall contain the current time in use by the device server represented as the count of the number of milliseconds elapsed since midnight, 1 January 1970 UT.

The LU initial policy access tag security attribute specifies the initial value for the policy access tag for a newly created logical unit. The initial value for this attribute shall be set to 7FFF FFFFh.

The policy access tag security attribute specifies the expected non-zero contents of the POLICY ACCESS TAG field in any capability that allows access to this logical unit.

Setting and querying security attributes are used by the application client by issuing the SECURITY PROTOCOL IN command (6.29) and SECURITY PROTOCOL OUT (6.30) command.

2.5.2 Security methods

2.5.2.1 Security methods overview

2.5.2 specifies how credentials are prepared and validated using the various security methods.

The list of supported security methods (see table xxx) applies to the device server. If the device server supports only logical unit based security attributes, the default security method is the security method assigned to a newly created logical unit. If the device server does not support only logical unit based security attributes, the security method is undefined for newly created logical units. The current security attribute may apply to the device server or to a particular logical unit.

This standard defines the following security methods:

- **NOSEC (No Security)**

  The device server does not verify the authenticity of the capability prior to performing an operation.

- **CAPKEY (Integrity of Capability (Access Control Security))**

  Access Control Security is based on the protocol presented and analyzed in [ACF+02]. CAPKEY provides a mechanism to prevent an application client from forging or otherwise modifying a credential or replaying a credential over a different authenticated channel. In addition, it verifies that the application client rightfully obtained the credential it is presenting.

The security methods are summarized in Table 9. All methods use the same basic flow and the same credential structure.

Table 9 - Security methods

Method	Description	Without a secure channel	With a secure channel
NOSEC	No security	No security	Network-level integrity
CAPKEY	Access control	End-to-end verification of credentials	+ Protection from network attacks

2.5.2.2 General

When the device server receives a command for a logical unit for which:

a) the CbCS bit is set in the standard Inquiry data (see 2.4);

b) the opcode field is set to 7Eh (i.e. Encapsulated SCSI Command CDB); and

c) the encapsulation type is 10h (i.e. PERFORM SECURE COMMAND CDB),

then the credential shall be validated before any other field in the CDB is validated.

If the security method for the logical unit is NOSEC, the credential is validated as described in 2.5.2.3.

If the Security method for the logical unit is CAPKEY, the credential is validated as described in 2.5.2.4.

When the device server receives a command for a logical unit for which:

a) the CbCS bit is set in the standard Inquiry data (see 2.4);

b) the opcode is other than 7Eh (i.e. not Encapsulated SCSI Command CDB); and

c) the received command requires authorization as described in 2.3,

then the command shall be terminated with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

When the device server receives a command for a logical unit for which:

a) the CbCS bit is set in the standard Inquiry data (see 2.4);

b) the opcode is 7Eh (i.e. Encapsulated SCSI Command CDB);

c) the value of the ENCAPSULATION TYPE field is other than 10h (i.e. not a PERFORM SECURE COMMAND CDB); and

d) the encapsulated command requires authorization as described in 2.3,

then the command shall be terminated with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

2.5.2.3 The NOSEC security method

2.5.2.3.1 Overview

The NOSEC security method validates that the capability authorizes the encapsulated command for each CDB.

The NOSEC method does not validate the integrity of the capability.

2.5.2.3.2 Preparing and validating credentials

Preparing credentials for the NOSEC security method does not require the knowledge of any secret information and may be done by the application client without coordination. Only the Capability part of the credential are specified, the Security Parameters are not used.

The device server validates the credential as follows:

1. Verify that the LU DESCRIPTION field in the CDB Capability, according to the LU DESCRIPTOR TYPE field and LU DESCRIPTOR LENGTH field, matches the logical unit for which the command is to be sent. If they don't match, the command shall be terminated with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.
2. If the POLICY ACCESS TAG field in the CDB Capability contains a non-zero value, compare it to the Policy Access Tag of the addressed logical unit. If they don't match, the command shall be terminated with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
3. Verify that the requested command in the REQUEST CDB field in the CDB is authorized by the PERMISSIONS BIT MASK field in the CAPABILITY descriptor in the CDB. If the requested command is not authorized, the command shall be terminated with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

2.5.2.4 The CAPKEY security method

2.5.2.4.1 Overview

The CAPKEY security method validates the integrity of the capability information in each PERFORM SECURE command (see 2.2). It provides comprehensive security when the service delivery subsystem between the device server and application client is secured via methods specified in the applicable SCSI transport protocol.

Given a credential and a channel, the protocol ties the message to the channel via a **validation tag**. The validation tag is computed by the client as $F_{C_{key}}^{pr}\left(ChannelID\right)$,

Where:

$ChannelID$ identifies the communication channel and is unique to this combination of initiator port, target port, and the particular I_T nexus on which they communicate.

C_{key} is the capability key associated with the command.

GP – stopped here

ChannelID is chosen by the target device; the protocol provides a mechanism for a client to request the value of the *ChannelID*. Given that the device server knows the channel on which a request was received and its *ChannelID*, the device server can verify that the validation tag attached to the request equals $F^{pr}_{C_{key}}(ChannelID)$. Note that the same validation tag can be used with all requests based upon a given credential.

This mechanism is used by the CAPKEY security method. Since the capability key C_{key} with which the validation tag $F^{pr}_{C_{key}}(ChannelID)$ is computed depends on the capability *CAP*, it ensures that the request is authenticated by a client who legally obtained the credential, whether directly form the security manager or indirectly via delegation.

Below are a number of performance considerations for the CAPKEY security method.

As long as the same channel is used,

- An application client does not need to request a new credential on every command; rather, the client can reuse the *CAP* and C_{key} on multiple commands for the same logical unit(s).
- The application client does not need to recalculate the validation tag on each command; rather, this needs to be calculated only once per credential.
- The device server does not need to recalculate C_{key} and $F^{pr}_{C_{key}}(ChannelID)$ on each exchange with a client. Rather since we assume a secure channel, these values need to only be calculated the first time the device server receives a given capability on a given channel, and then the computed value can be cached and reused in processing every command on the channel.

2.5.2.4.2 Preparing and validating credentials

The application client computes the CDB REQUEST INTEGRITY CHECK VALUE field contents using:

a) The algorithm specified in the capability INTEGRITY CHECK VALUE ALGORITHM field;

b) The security token returned in the CbCS Attributes page (see 6.29.3.5); and

c) The credential capability key (which shall be generated by the security manager).

The device server validates the credential as follows:

1. Compute the Capability-Key, using:
    a. The algorithm specified in the INTEGRITY CHECK VALUE ALGORITHM field in the capability
    b. All the bytes of the Capability field of the CDB
    c. The secret key as follows:

   - i. If the encapsulated command is SECURITY PROTOCOL OUT command specifying the CbCS security protocol and the Set Key page – the authentication master key.
   - ii. If the encapsulated command is SECURITY PROTOCOL OUT command specifying the CbCS security protocol and the Set Master Key, Seed Exchange page – the authentication master key.
   - iii. If the encapsulated command is SECURITY PROTOCOL IN command specifying the CbCS security protocol and the Set Master Key, Seed Exchange page – the authentication master key.
   - iv. If the encapsulated command is SECURITY PROTOCOL OUT command specifying the CbCS security protocol and the Set Master Key, Change Master Key page – the next authentication master key computed after GOOD status has been returned by the SECURITY PROTOCOL IN command specifying the CbCS security protocol and the Set Master Key, Seed Exchange page.
   - v. For any other encapsulated command – the authentication working key:
     1. Identified by the KEY VERSION field in the Capability; and
     2. Associated with the target LU, or if doesn't exist (or invalid) in the target LU and the device server supports global keys (see 6.29.3.4) – associated with the well-known SECURITY PROTOCOL LU.

2. Compute the integrity check value, using:
   - a. The algorithm specified in the INTEGRITY CHECK VALUE ALGORITHM field in the capability
   - b. The security token returned in the CbCS Attributes page (see 6.29.3.5)
   - c. The Capability-Key as the key
3. Compare the integrity check value computed in the previous step with the REQUEST INTEGRITY CHECK VALUE in the CDB. If they don't match, the command shall be terminated with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.
4. Validate that the capability matches the target logical unit and that the capability authorizes the encapsulated command as described in 2.5.2.3.2.

Credential validation as described above should be performed before any validation of any of the encapsulated CDB fields is done.

2.5.3 Integrity check value algorithms

Table 10 specifies coded values to be used by application clients and device servers to specify the algorithm used to compute integrity check values. These algorithms are typically pseudo-random functions.

Table 10 - Integrity check value algorithms

value	Algorithm	Reference
0000h	Reserved	
0001h	HMAC-SHA-1	FIPS 180-2 (2002) and FIPS 198 (2002)
0002h	HMAC-SHA-256	FIPS 180-2 (2002) and FIPS 198 (2002)
0003h	HMAC-SHA-384	FIPS 180-2 (2002) and FIPS 198 (2002)
0004h	HMAC-SHA-512	FIPS 180-2 (2002) and FIPS 198 (2002)
0005h	PRF_AES128_CBC	RFC 4434
0006h – 0FFFh	Reserved	
1000h – FFFFh	Vendor specific	

Note: This table is partially based on table K3 in 06-449r1. We should consider merging the two tables into one in a common section (cryptography section?) that both IKEv2-SCSI and CbCS will reference.

2.5.4 DH group identifiers

Table 11 shows the valid Diffie-Hellman algorithm identifiers (i.e., group identifiers) for IKEv2-SCSI. In Table 11, the column entitled "Key Size" indicates the size, in bytes, of the DEVICE SERVER DH_DATA field in the SEED EXCHANGE phase of the SET MASTER KEY command (2.6.3). A device server should not support finite field Diffie-Hellman groups with less that 2048 bits or elliptic curve fields of less than 256 bits.

Table 11 - Diffie-Hellman group identifiers

Value	Description	Key Size	Reference
0000h – 000Ch	Restricted		IANA
000Dh	2048-bit MODP group (finite field D-H)	256	RFC 3526
000Eh	3072-bit MODP group (finite field D-H)	384	RFC 3526
000Fh – 0012h	Restricted		IANA
0013h	256-bit prime elliptic curve field P-256	32	NIST FIPS 186-2
0014h	384-bit prime elliptic curve field P-384	48	NIST FIPS 186-2
0015h	521-bit prime elliptic curve field P-521	66	NIST FIPS 186-2
0016h – 03FFh	Restricted		IANA
0400h – 0FFFh	Reserved		
1000h - FFFFh	Vendor specific		

Note: This table is based on table K5 in T10/06-449r1. We should consider merging the two tables into one in a common section `(cryptography section?) that both IKEv2-SCSI and CbCS will reference.`

2.5.5 Setting and querying security attributes

Changes in section 6.29 SECURITY PROTOCOL IN command

6.29.1 SECURITY PROTOCOL IN command description

Table 186 — SECURITY PROTOCOL field in SECURITY PROTOCOL IN command

Code	Description	Reference
00h	Security protocol information	6.29.2
01h - 06h	Defined by the TCG	3.1.128
07h	Capability based Command Security	
08h - 1Fh	Reserved	
20h	Tape Data Encryption	SSC-3
21h - EDh	Reserved	
EEh	Authentication in Host Attachments of Transient Storage Devices	IEEE P1667
EFh	ATA Device Server Password Security	TBD
F0h - FFh	Vendor Specific	

New section:

6.29.3 Capability based Command Security (CbCS)

6.29.3.1 Overview

The SECURITY PROTOCOL IN command specifying 'Capability based Command Security' security protocol requests the device server to return information about the security attributes of the logical unit, or the security attributes of the device server applying to ay logical unit – if targeted to the well-known SECURITY PROTOCOL logical unit. The command supports a series of pages that are requested individually. An application client requests a page by using a SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to 'Capability based Command Security' and the SECURITY PROTOCOL SPECIFIC field set to the page code requested.

A device server that supports the 'Capability based Command Security security' protocol in the SECURITY PROTOCOL OUT command shall also support a SECURITY PROTOCOL IN command specifying the 'Capability based Command Security' protocol.

The SECURITY PROTOCOL SPECIFIC field (see Table 12) specifies the type of report that the application client is requesting.

Table 12 – SECURITY PROTOCOL SPECIFIC field values

Code	Description	Support	Reference
0000h	CbCS In Support page	M	6.29.3.2

Code	Description	Support	Reference
0001h	CbCS Out Support page	M	6.29.3.3
0002h-000Fh	Reserved		
0010h	CbCS Capabilities page	M/O (?)	6.29.3.4
0011h	CbCS attributes page	M	6.29.3.5
0012h	Set Master Key, Seed Exchange	M	6.29.3.6
0013h – FEFFh	Reserved		
FF00h-FFFFh	Vendor specific		
Support key: M – Mandatory for device servers that support the CbCS protocol O – Optional			

6.29.3.2 CbCS In Support page

Table 13 specifies the format of the CbCS In Support page.

Table 13 - CbCS In Support page format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0000h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	(MSB)	CbCS In Support page code (first)						
5								(LSB)
	⋮							
n-1	(MSB)	CbCS In Support page code (last)						
n								(LSB)

The CbCS In Support page shall contain a list of all of the pages that the device server supports for the SECURITY PROTOCOL IN command specifying the 'Capability based Command Security' security protocol in ascending order beginning with page code 0000h.

6.29.3.3 CbCS Out Support page

Table 14 specifies the format of the CbCS Out Support page.

Table 14 - CbCS Out Support page format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0001h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	(MSB)	CbCS Out Support page code (first)						
5								(LSB)
	. . .							
n-1	(MSB)	CbCS Out Support page code (last)						
n								(LSB)

The CbCS Out Support page shall contain a list of all of the pages that the device server supports for the SECURITY PROTOCOL OUT command specifying the 'Capability based Command Security' security protocol in ascending order.

6.29.3.4 CbCS Capabilities page

Table 15 specifies the format of the CbCS Capabilities page.

Table 15 – CbCS Capabilities page format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0010h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (i*2+j*2+k*2+8)						
3								(LSB)
4	GKS	LUKS	GSMS	LUSMS	Reserved			
5	Reserved							
6		Number of supported security methods (i)						
7								(LSB)
8		SUPPORTED SECURITY METHOD (first)						
9								(LSB)
	. . .							

Byte \ Bit	7	6	5	4	3	2	1	0
i*2+6	SUPPORTED SECURITY METHOD (last)							
i*2+7								(LSB)
i*2+8	Number of supported integrity check value algorithms (j)							
i*2+9								(LSB)
i*2+10	SUPPORTED INTEGRITY CHECK VALUE ALGORITHM (first)							
i*2+11								(LSB)
	. . .							
i*2+j*2+8	SUPPORTED INTEGRITY CHECK VALUE ALGORITHM (last)							
i*2+j*2+9								(LSB)
i*2+j*2+10	Number of supported D-H groups (k)							
i*2+j*2+11								(LSB)
i*2+j*2+12	SUPPORTED D-H GROUP (first)							
i*2+j*2+13								(LSB)
	. . .							
i*2+j*2+k*2+10	SUPPORTED D-H GROUP (last)							
i*2+j*2+k*2+11								(LSB)

A Global Keys Support (GKS) bit set to one specifies that the device server supports global keys. A Global Keys Support (GKS) bit set to zero specifies that the device server does not support global keys.

A Logical Unit Keys Support (LUKS) bit set to one specifies that the device server supports per-LU keys. A Logical Unit Keys Support (LUKS) bit set to zero specifies that the device server does not support per-LU keys.

A device server that supports CbCS must support either global keys or per-LU keys, and it may support both.

A Global Security Method Support (GSMS) bit set to one specifies that the device server supports global security method (applied to any LU). A Global Keys Support (GKS) bit set to zero specifies that the device server does not support global security method (security methods must be assigned to each LU in particular).

A Logical Unit Security Method Support (LUSMS) bit set to one specifies that the device server supports per-LU security method. A Logical Unit Keys Support (LUSMS) bit set to zero specifies that the device server does not support per-LU security method.

A device server that supports CbCS must support either global security method or per-LU security method, and it may support both.

Each SUPPORTED SECURITY METHOD field contains a coded value of a security method supported by the device server (see Table 4 - Security method).

Each SUPPORTED INTEGRITY CHECK VALUE ALGORITHMS field contains a coded value of an algorithm to compute integrity check values supported by the device server, according to the coded values specified in Table 10.

The SUPPORTED DH GROUP attributes contain coded values identifying the supported values in the DH_GROUP field of a SET MASTER KEY command (see 2.6.3).

6.29.3.5 CbCS Attributes page

Table 16 specifies the format of the CbCS Attributes page.

Table 16 - CbCS Attributes page format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0010h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	(MSB)	SECURITY METHOD						
5								(LSB)
6	(MSB)	POLICY ACCESS TAG						
9								(LSB)
10		MASTER KEY IDENTIFIER						
17								(LSB)
18		WORKING KEY IDENTIFIER 0						
25								(LSB)
	. . .							
138		WORKING KEY IDENTIFIER 15						
145								(LSB)
146	(MSB)	CLOCK						
151								(LSB)
152	Reserved							
153	SECURITY TOKEN LENGTH							
154	SECURITY TOKEN							

Bit / Byte	7	6	5	4	3	2	1	0
n								(LSB)

If the target LU is the well-known security protocol LU, the SECURITY METHOD field is the security method assigned by the device server to a newly created logical unit. If the target LU is not the well-known security protocol LU, the SECURITY METHOD field is the current security method used for the device. If the target LU is the well-known security protocol LU and the device server does not support global security method (i.e., the GSMS bit is set to zero in the CbCS Capabilities page) then this field is undefined. If the target LU is not the well-known security protocol LU and the device server does not support per-LU security method (i.e., the LUSMS bit is set to zero in the CbCS Capabilities page) then this field is undefined. Security methods are described in detail in 2.5.2.

If the target LU is the well-known security protocol LU, the POLICY ACCESS TAG field is the initial policy access tag assigned by the device server to a newly created logical unit. If the target LU is not the well-known security protocol LU, the POLICY ACCESS TAG field is the current policy access tag assigned to the logical unit.

Keys may not be supported for the target LU. If the device server does not support global keys and the target LU is the well-known security protocol, the MASTER KEY IDENTIFIER field and all the WORKING KEY IDENTIFIER fields should contain FFFF FFFFh. If the device server does not support per-LU keys and the target LU is not the well-known security protocol, the MASTER KEY IDENTIFIER field and all the WORKING KEY IDENTIFIER fields should contain FFFF FFFFh. If keys are supported for the target LU, the values of those fields are as follows.

The MASTER KEY IDENTIFIER field contains the key identifier value from the most recent successful SET MASTER KEY command (see 2.6.3). If a SET MASTER KEY command has never been processed, the MASTER KEY IDENTIFIER field shall contain FFFF FFFE.

Each KEY IDENTIFIER field contains the key identifier value from the most recent successful SET KEY command with the KEY VERSION field set to the pertinent key (0-15) (see 2.6). If a key is invalid (i.e., never set, invalidated by a SET MASTER KEY command, or invalidated by a SET KEY command), the pertinent KEY IDENTIFIER field should contain 0000 0000h.

The CLOCK field shall contain the current time in use by the device server represented as the count of the number of milliseconds elapsed since midnight, 1 January 1970 UT.

Note: The clock field may be usable for other purposes. Perhaps it can be moved to a more generic place, e.g. mode page…

For the CAPKEY security method, the SECURITY TOKEN field contains a value that is unique to the I_T or the I_T_L nexus that sent the SECURITY PROTOCOL IN command. The security token shall be random as defined by RFC 1750. An I_T nexus loss event, logical unit reset event, or reset event (see SAM-4) shall cause the security token to change.

6.29.3.6 Set Master Key, Seed Exchange

Table 17 specifies the format of the Set Master Key, Seed Exchange page.

Table 17 - Set Master Key, Seed Exchange format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0012h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	DH DATA							
n								

If a SECURITY PROTOCOL IN command specifying this page is received and no SECURITY PROTOCOL OUT command specifying Set Master Key, Seed Exchange page has been received on the same I_T_L nexus during the past ten seconds, the command shall be terminated with a CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN CDB.

A device server that receives a SECURITY PROTOCOL OUT command specifying Set Master Key, Seed Exchange page on one I_T_L nexus may terminate the command with a CHECK CONDITION status, setting the sense key to ILLEGAL REQUEST and the additional sense code to SYSTEM RESOURCE FAILURE if either of the following command processing is incomplete on a different I_T_L nexus:

1) SECURITY PROTOCOL OUT command specifying Set Master Key, Seed Exchange page (6.30.1.4)
2) SECURITY PROTOCOL IN command specifying Set Master Key, Seed Exchange page (6.29.3.6)
3) SECURITY PROTOCOL OUT command specifying Set Master Key, Change Master Key page (6.30.1.5)

The DH DATA field contains the device server DH_data computed as follows:

1) A random number, y, is generated having a value between 0 and DH_prime minus one observing the requirements in RFC 1750; and
2) The device server DH_data is equal to DH_generator^{y} modulo DH_prime, where the DH_generator and DH_prime values are identified by the code value in the CDB DH_GROUP field.

After GOOD status has been returned for this command and before the SECURITY PROTOCOL OUT command specifying the Change Master Key page is processed, the next authentication master key and next generation master key shall be computed as described in 6.30.1.6 (see 2.6.4), using a seed value that is the concatenation of the following:

1) DH_generator^{xy} modulo DH_prime;

2) The Device Identification VPD page (83h) returned from the target LU for the INQUIRY command (see 7.6.3)

Changes in 6.30 SECURITY PROTOCOL OUT command:

6.30 SECURITY PROTOCOL OUT command

<Unchanged text here>

The SECURITY PROTOCOL field (see table 191) specifies which security protocol is being used.

Table 191 — SECURITY PROTOCOL field in SECURITY PROTOCOL OUT command

Code	Description	Reference
00h	Reserved	
01h - 06h	Defined by the TCG	3.1.128
07h	Capability based Command Security	
08h - 1Fh	Reserved	
20h	Tape Data Encryption	SSC-3
21h - EDh	Reserved	
EEh	Authentication in Host Attachments of Transient Storage Devices	IEEE P1667
EFh	ATA Device Server Password Security	TBD
F0h - FFh	Vendor Specific	

New section:

6.30.1 Capability based Command Security (CbCS)

6.30.1.1 Overview

The SECURITY PROTOCOL OUT command specifying 'Capability based Command Security' security protocol is used to configure the CbCS attributes in the device server. The command supports a series of pages that are sent individually. An application client requests to send a page by using a SECURITY PROTOCOL OUT command with the SECURITY PROTOCOL field set to 'Capability based Command Security' security protocol and the SECURITY PROTOCOL SPECIFIC field set to the page code requested.

The SECURITY PROTOCOL SPECIFIC field (see Table 18) specifies the type of page that the application client is sending.

Table 18 - SECURITY PROTOCOL SPECIFIC field values

Code	Description	Support	Reference
0000h – 0010h	Reserved		
0011h	Set CbCS Attributes	O	6.30.1.2

Code	Description	Support	Reference
0012h	Set Key	M	6.30.1.3
0013h	Set Master Key, Seed Exchange	M	6.30.1.4
0014h	Set Master Key, Change Master Key	M	6.30.1.5
0015h – FEFFh	Reserved		
FF00h-FFFFh	Vendor specific		
Support key: M – Mandatory for device servers that support the CbCS protocol O – Optional			

If the SECURITY PROTOCOL SPECIFIC field is set to a reserved or unsupported value, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

6.30.1.2 Set CbCS attributes page

Table 19 specifies the format of the Set CbCS Attributes page.

Table 19 – Set CbCS Attributes page format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0011h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (84)						
3								(LSB)
4	(MSB)	SECURITY METHOD						
5								(LSB)
6	(MSB)	POLICY ACCESS TAG						
9								(LSB)
10	Capability							
67								
68	Security Parameters							
87								

The PAGE LENGTH field indicates the number of bytes of parameter data to follow. If the page length value results in the truncation of any field, the device server shall terminate the command with CHECK CONDITION status, with the sense key set to

ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The SECURITY METHOD field specifies a new security method to apply to the target LU (see 2.5.2). The reserved value FFFFh can be used by the application client to specify that it does not request to change the current security method. If the SECURITY METHOD field contains a value that is equal to the current security method, it should not be treated as an error. If the specified value does not match any of the supported security methods reported in the CbCS Capabilities page (see 6.29.3.4), the device server shall terminate the command with CHECK CONDITION status and set the sense key to ILLEGAL REQUEST and the additional sense code to INVALID FIELD IN PARAMETER DATA.

The POLICY ACCESS TAG field specifies a new policy access tag for the target LU. The reserved value of FFFF FFFFh can be used by the application client to specify that it does not request to change the current policy access tag value. If the target LU is the well-known SECURITY PROTOCOL LU and the POLICY ACCESS TAG field contains any value other than FFFF FFFFh, the device server shall terminate the command with CHECK CONDITION status and set the sense key to ILLEGAL REQUEST and the additional sense code to INVALID FIELD IN PARAMETER DATA.

The device server shall validate the capability as described in 2.5.2 according to the current security method applied to the target LU (not the security method specified in this page). The Capability format is described in **Error! Reference source not found.**. The SEC MGMT bit must be set to one in the Permissions Bit Mask field of the Capability. No other bit is required in the Permissions Bit Mask field for this command. The Security Parameters format is described in **Error! Reference source not found.**.

2.6 Key management

2.6.1 Overview

All credentials are based on a secret key that is shared between the device server and the security manager. To prevent an adversary from obtaining too many credentials generated with the same key, keys must be refreshed regularly.

Key management requirements are:

- The security manager should be able to replace the target device keys in a secure manner even if the channel it has with the target device is not secure.
- The target device must support multiple keys. This allows the security manager to set the keys in a manner that their durations overlap.
- A random source to generate keys is required only from the manager, and not from the target device.
- A target device crash or loss of a logical unit should not necessarily invalidate valid credentials.

The keys are used to generate the capability keys used by clients to access individual logical units. Because of their frequent use working keys should be refreshed very frequently, e.g., on an hourly basis. Unfortunately, a key refresh immediately invalidates all credentials generated by that key. This could result in significant performance degradation as all the clients would be required to communicate with the security manager in order to get new credentials. The load on the target device would also increase because all new credentials would have to be explicitly validated before being cached by the device server. To mitigate these problems, the target device may declare multiple (up to 16) refreshed versions of the key as valid. This effectively defines multiple keys that are valid concurrently. Therefore, a key refresh would impact a limited number of capabilities. To support this feature, the protocol defines a key version field that is incorporated in the capability indicating which key should be used in the validation process.

The number of active key versions used is agreed between the device server and the security manager. When setting a new key, the security manager tags the key with a version number (between 0 and 15); the device server uses this tag to determine which key to use in validating a command.

A generation master key is used to generate working keys. An authentication master key is used to generate credentials for commands to set and refresh keys. The master key can also be refreshed. Refreshing the master key is done by a Diffie-Hellman key exchange algorithm carried over a sequence of three commands (see 2.6.3).

Separate set of master key and working keys may be used for each logical unit, or a single set may be used for all logical units served by a device server (using the well-known security protocol logical unit). The standard supports three options:

1. A single set of master key and working keys is used by the device server through the well-known security protocol logical unit. This set of keys serves all the logical units within the device.
2. A separate set of master key and working keys is used for each logical unit within the device.
3. A single set of master key and working keys is used by the device server through the well-known security protocol logical unit. In addition, a separate set of master key and working keys **may be** used for any logical unit within the device. The global set of keys serves any logical unit that doesn't have its own set of keys.

The standard does **not** support the following features:

1. A single set of keys serving a distinct set of logical units. Supporting this feature would have required adding the notion of logical unit sets in the device server. Note that a security manager can still use the same keys for a set of logical units, but this is transparent to the device server.
2. A global master key s used to generate logical unit specific working keys. This feature doesn't seem to give significant capabilities on top of a single set of working keys.

2.6.2 Setting working keys

6.30.1.3 Set Key page

Table 20 specifies the Set Key page format.

Table 20 - Set Key page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0012h)						

Byte \ Bit	7	6	5	4	3	2	1	0
1								(LSB)
2	(MSB)			PAGE LENGTH (30)				
3								(LSB)
4				Reserved				
5		Reserved				KEY VERSION		
6	(MSB)			KEY IDENTIFIER				
13								(LSB)
14	(MSB)			SEED				
33								(LSB)

The KEY VERSION field specifies the working key version to be updated.

The KEY IDENTIFIER field specifies a unique identifier to be associated with the new key. Successful processing of the SET KEY command shall include storing the key identifier value in the attribute specified in the CbCS Attributes page (6.29.3.5).

The SEED field contains a random number generated from a good source of entropy (e.g., as described in RFC 1750). The updated key value shall be computed as described in 6.30.1.5.

The device server shall validate the capability as described in 2.5.2 according to the current security method applied to the target LU (not the security method specified in this page). The Capability format is described in **Error! Reference source not found.**. The SEC MGMT bit must be set to one in the Permissions Bit Mask field of the Capability. No other bit is required in the Permissions Bit Mask field for this command. The Security Parameters format is described in **Error! Reference source not found.**.

2.6.3 Setting the master key

Setting the master key is a complex operation involving Diffie-Hellman key exchange. It provides forward secrecy for the master key. That is, the disclosure of a master key does not compromise any other master key derived in the series of master key generations.

The operation requires a series of three commands:

1) SECURITY PROTOCOL OUT command specifying Set Master Key, Seed Exchange page (6.30.1.4)
2) SECURITY PROTOCOL IN command specifying Set Master Key, Seed Exchange page (6.29.3.6)
3) SECURITY PROTOCOL OUT command specifying Set Master Key, Change Master Key page (6.30.1.5)

6.30.1.4 Set Master Key, Seed Exchange page

Table 21 specifies the Set Master Key, Seed Exchange page format.

Table 21 - Set Master Key page format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0013h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	(MSB)	DH GROUP						
5								(LSB)
6	(MSB)	DH DATA LENGTH						
9								(LSB)
10	DH DATA							
n								

A device server that receives a SECURITY PROTOCOL OUT command specifying Set Master Key, Seed Exchange page on one I_T_L nexus may terminate the command with a CHECK CONDITION status, setting the sense key to ILLEGAL REQUEST and the additional sense code to SYSTEM RESOURCE FAILURE if either of the following command processing is incomplete on a different I_T_L nexus:

3) SECURITY PROTOCOL OUT command specifying Set Master Key, Seed Exchange page (6.30.1.4)
4) SECURITY PROTOCOL IN command specifying Set Master Key, Seed Exchange page (6.29.3.6)
5) SECURITY PROTOCOL OUT command specifying Set Master Key, Change Master Key page (6.30.1.5)

The DH GROUP field contains the coded value of the Diffie-Hellman group (see 2.5.4) that identifies the DH_generator and DH_prime values to be used in the seed exchange. If the value in the DH GROUP field does not appear in one of the SUPPORTED D-H GROUP fields in the CbCS Capabilities page (see 6.29.3.4) the command shall be terminated with a CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN CDB.

The DH DATA LENGTH field specifies the number of bytes of application client DH_data in the DH DATA field. The application client DH_data is computed as follows:

1) A random number, x, is generated having a value between 0 and DH_prime minus one observing the requirements in RFC 1750; and
2) The application client DH_data is equal to DH_generator^x modulo DH_prime, where the DH_generator and DH_prime values are identified by the code value in the DH GROUP field.

6.30.1.5 Set Master Key, Change Master Key page

Table 22 specifies the format of the Set Master Key, Change Master Key page.

Table 22 - Set Master Key, Change Master Key page format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)	PAGE CODE (0014h)						
1								(LSB)
2	(MSB)	PAGE LENGTH (n-3)						
3								(LSB)
4	(MSB)	KEY IDENTIFIER						
11								(LSB)
6	(MSB)	APPLICATION CLIENT DATA LENGTH (k-9)						
9								(LSB)
10	APPLICATION CLIENT DH DATA							
k								
k+1	(MSB)	DEVICE SERVER DATA LENGTH (n-(k+4))						
k+4								(LSB)
k+5	DEVICE SERVER DH DATA							
n								

The KEY IDENTIFIER field specifies a unique identifier to be associated with the new master key. Successful processing of the SECURITY PROTOCOL OUT command specifying Set Master Key, Change Master Key page - shall include storing the key identifier value in the MASTER KEY IDENTIFIER field in the CbCS Attributes page (see 6.29.3.5).

Table 23 specifies special key identifiers that shall not be used by the application client when setting keys. Any value not listed in the table is permitted for use by the application client when setting keys.

Table 23 - Special key identifiers

Value	Description
0000-0000h	This value denotes a key that was never set or was invalidated
FFFF-FFFEh	This value denotes an initial key set by the device server
FFFF-FFFFh	This value denotes that the pertinent key is not supported by the device server

The command shall be terminated with a CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to

INVALID FIELD IN PARAMETER LIST if the value of the KEY IDENTIFIER field contains a value that is listed in Table 23.

The APPLICATION CLIENT DATA LENGTH field specifies the number of bytes that follow in the APPLICATION CLIENT DH DATA field.

The APPLICATION CLIENT DH_DATA field contains the application client DH_data from the last SECURITY PROTOCOL OUT command specifying Set Master Key, Seed Exchange page on the same I_T_L nexus.

The DEVICE SERVER DATA LENGTH field specifies the number of bytes that follow in the DEVICE SERVER DH DATA field.

The DEVICE SERVER DH DATA field contains the device server DH_data from the last SECURITY PROTOCOL IN command specifying Set Master Key, Seed Exchange page on the same I_T_L nexus.

The command shall be terminated with a CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST if the parameter data fails to compare in any of the following ways with the data exchanged in the SECURITY PROTOCOL OUT command specifying Set Master Key, Seed Exchange page and in the SECURITY PROTOCOL IN command specifying Set Master Key, Seed Exchange page that were most recently processed on this I_T_L nexus since a I_T nexus loss event, logical unit reset event, or reset event (see SAM-4), if any:

a) The contents of the APPLICATION CLIENT DATA LENGTH field do not match the contents of the DH DATA LENGTH field in the SECURITY PROTOCOL OUT Set Master Key, Seed Exchange page;

b) The contents of the APPLICATION CLIENT DH DATA field do not match the contents of the DH DATA field in the SECURITY PROTOCOL OUT Set Master Key, Seed Exchange page;

c) The contents of the DEVICE SERVER DATA LENGTH field do not match the contents of the PAGE LENGTH field in the SECURITY PROTOCOL IN Set Master Key, Seed Exchange page; or

d) The contents of the DEVICE SERVER DH DATA field do not match the contents of the DH DATA field in the SECURITY PROTOCOL IN Set Master Key, Seed Exchange page.

Successful processing of a SECURITY PROTOCOL OUT command specifying Set Master Key, Change Master Key page shall:

a) Replace the authentication master key with the next authentication master key computed after the return of GOOD status for the most recent SECURITY PROTOCOL IN command specifying Set Master Key, Seed Exchange page (see 6.29.3.6);

b) Replace the generation master key with the next generation master key computed after the return of GOOD status for the most recent SECURITY PROTOCOL IN command specifying Set Master Key, Seed Exchange page;

c) Invalidate all the working keys on the logical unit.

For every key that is invalidated by a SET MASTER KEY command CHANGE MASTER KEY step, the associated key identifier attribute shall have its attribute set to 0000 0000h.

Note: The next authentication master key computed after the return of GOOD status for the most recent SECURITY PROTOCOL IN command specifying the Set Master Key, Seed Exchange page (see 6.29.3.6) shall be used to compute the capability key for the SECURITY PROTOCOL OUT command specifying Set Master Key, Change Master Key page (see 2.5.2.4).

2.6.4 Computing new and updated keys

6.30.1.6 Computing updated generation keys and new authentication keys

When processing the Set Key page (6.30.1.3) and the Set Master Key pages (6.30.1.4, 6.29.3.6, and 6.30.1.5), the device server shall perform the steps described in this subclause to compute new generation and authentication keys.

The inputs to the process are:

a) The input key value is one of the following:

  A) For a Set Key page, the master key generation key; or

  B) For a Set Master Key page, the previous master key generation key shall be used;

b) The seed value is one of the following:

  A) For a Set Key page, the contents of the SEED field of the Set Key page; or

  B) For a Set Master Key, Change Master Key page, the value computed after GOOD status has been returned for the SECURITY PROTOCOL IN command specifying the Set Master Key, Seed Exchange page (see 6.29.3.6) and updated by the Set Master Key, Changed Master Key page (see 6.30.1.5);

and

c) The integrity check value algorithm, as specified in the INTEGRITY CHECK VALUE ALGORITHM field in the capability in the pertinent page.

2.7 RECEIVE CREDENTIAL command

A new sub-section in section 6:

6.X RECEIVE CREDENTIAL command

The RECEIVE CREDENTIAL command (see **Error! Reference source not found.**) provides a means for the application client to receive a credential for use in the PERFORM SECURE COMMAND command (see 2.2).

Table 24 - RECEIVE CREDENTIAL CDB format

Byte \ Bit	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							

Bit / Byte	7	6	5	4	3	2	1	0
6								
7	ADDITIONAL CDB LENGTH (n-7)							
8	(MSB)	SERVICE ACTION (1800h)						
9								(LSB)
10	(MSB)	ALLOCATION LENGTH						
11								(LSB)
12	Reserved				LU DESCRIPTOR TYPE			
13	LU DESCRIPTOR LENGTH							
14	LU DESCRIPTOR							
n								

If a RECEIVE CREDENTIAL command is received and no SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to IKEv2-SCSI and the SECURITY PROTOCOL SPECIFIC field in the CDB set to 0103h has completed successfully on the same I_T_L nexus, the command shall be terminated with a CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN CDB.

`Note: The above paragraph is based on 06-449r1 and depends on it.`

The ALLOCATION LENGTH field specifies the number of bytes available to receive the credential (see Table 26) sent in response to the RECEIVE CREDENTIAL command. If the allocation length is not sufficient to contain credential, the command shall be terminated with a CHECK CONDITION status, the sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN CDB.

The LU DESCRIPTOR TYPE field (see Table 25) specifies the format of information that appears in the LU DESCRIPTOR field.

Table 25 - LU DESCRIPTOR TYPE codes

Code	Description
0h	Vendor specific
1h – 2h	Reserved
3h	NAA
4h – Fh	Reserved

The specification of the format of the descriptor types listed in Table 25 is defined in section 7.6.3 (Device Identification VPD page).

In order to implement credential based on logical unit unique identifier, the same identifier type shall be 1) used in the access control policies set in the policy manager; 2) returned by the device server in Device Identification VPD page (Inquiry page 83h); 3) used by the application client to identify the device and request the corresponding credential from the security manager.

The LU DESCRIPTOR LENGTH field indicates the length in bytes of the LU DESCRIPTOR field. If the value of this field is larger than 16, the command shall be terminated with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

In response to the RECEIVE CREDENTIAL CDB, the device server sends a credential in the format specified in Table 26.

Table 26 - Credential format

Bit / Byte	7	6	5	4	3	2	1	0
0	CREDENTIAL FORMAT (1h)				Reserved			
1	Reserved							
2	(MSB)	CREDENTIAL LENGTH (n-3)						
3								(LSB)
4	(MSB)	CAPABILITY LENGTH (k-5)						
5								(LSB)
6	Capability							
k								
k+1	(MSB)	CAPABILITY KEY LENGTH (n-(k+4))						
k+4								(LSB)
k+5	CAPABILITY KEY							
n								

The CAPABILITY LENGTH specifies the length of the Capability field.

The format of the Capability is defined in **Error! Reference source not found.**.

CAPABILITY KEY LENGTH specifies the length of the CAPABILITY KEY field.

The CAPABILITY KEY field contains a value that the application client shall use for preparing credentials as described in 2.5.

2.8 TODO

- Modify Table C.8 — Variable Length CDB Service Action Codes Used by All Device Types.
- Modify Table 48 — Commands for all device types.

3 Changes to SBC-3

Proposal 07-029 defines ESC (Encapsulated SCSI Command) CDB format. This proposal is dependent on 07-029r1.

3.1 ENCAPSULATION TYPE definitions

Table 27 lists the ENCAPSULATION TYPE codes defined in this standard for use with ESC CBD format (see section 4.3.4 in SPC-4).

Table 27 - OUTERMOST ENCAPSULATION TYPE field and NEXT ENCAPSULATION TYPE field

		Size of encapsulation parameter descriptors (bytes)		
Code	**Description**	**Prefix**	**Postfix**	**Reference**
Prefix and Postfix codes				
40h – 47h	Reserved			
Prefix only codes				
48h	Specify LBA Range	16	0	
49h – 5Fh	Reserved		0	

3.2 PERFORM SECURE COMMAND ON LBA RANGE

The PERFORM SECURE COMMAND ON LBA RANGE command (see **Error! Reference source not found.**) allows applying security to an implemented SBC-3 command using the parameters specified in the encapsulation parameters fields.

The PERFORM SECURE COMMAND ON LBA RANGE command is designated by the combination of OPERATION CODE 7Eh, OUTERMOST ENCAPSULATION TYPE 48h, and NEXT ENCAPSULATION TYPE 10h.

Table 28 - PERFORM SECURE COMMAND ON LBA RANGE format

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Eh)							
1	DATA TRANSFER		Reserved					
2	Reserved							
3	OUTERMOST ENCAPSULATION TYPE (48h)							
4	NEXT ENCAPSULATION TYPE (10h)							
5	Reserved							
6	(MSB)	LBA START						

Bit / Byte	7	6	5	4	3	2	1	0
13								(LSB)
14	(MSB)			LENGTH				
21								(LSB)
22	NEXT ENCAPSULATION TYPE (0)							
23	Reserved							
24 – 81	Capability							
82 – 121	Security Parameters							
122 – n	Encapsulated CDB							

The encapsulation parameters – LBA START and LENGTH specify the LBA range within the logical unit to which the Capability applies.

If the value of the LENGTH field is 0, the LBA range spans from the specified LBA START to the last LBA of the logical unit. Otherwise, the LBA range spans from 'LBA START' to 'LBA START' + 'LENGTH' - 1.

The Capability format is described in **Error! Reference source not found.**.

The encapsulation parameters (the LBA START and LENGTH) should be appended to the Capability for the purpose of preparing the Security Parameters as defined for the security method (see 2.5.2) specified in the Capability.

The Security Parameters are as described in **Error! Reference source not found.**.

The encapsulated CDB may be any CDB defined in this standard command set.

The device server shall perform the following validation after validating the command as defined for the security method specified in the Capability:

Compare the LBA range specified by the LOGICAL BLOCK ADDRESS and LENGTH fields in the CDB Capability to the LBA range affected by the encapsulated command CDB. If the LBA range in the Capability does not fully cover the LBA range affected by the encapsulated command, the command shall be terminated with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

Should we specify permissions bit masks for SBC-3 commands in SBC-3?

4 Issues

- The OSD standard says that when security method other than NOSEC is used, reservation commands (including persistent reservations) shall be treated as invalid. Should this apply to CbCS? What is the impact on working systems?
- The OSD standard says:

  If the root object or any partition in the OSD logical unit is using any security method other than NOSEC, all SAM-3 task management functions except QUERY TASK shall be ignored and responded to as if they have been successfully processed. The PERFORM TASK MANAGEMENT FUNCTION command (see 6.16) allows SAM-3 task management functions to be processed under the protection of the current security method.

  Should this apply to CbCS? What is the impact on working systems?

5 References

[ObsSec05] Michael Factor, David Nagle, Dalit Naor, Erik Riedel, and Julian Satran. The OSD Security Protocol, September 2005.
http://ieeeia.org/sisw/2005/PreProceedings/04.pdf

[OSD04] R. O. Weber. SCSI Object-Based Storage Device Commands – 2 (OSD-2), Date: 2004/10/04, Rev: 00. InterNational Committee for Information Technology Standards (formerly NCITS), October 2004.
http://www.t10.org/drafts.htm.

[BCK96] M. Bellare, R. Canetti, and H. Krawczyk. Message authentication using hash functions: The hmac construction. RSA Laboratories' Crypto-Bytes, 2(1), Spring 1996.

[FIPS180-02] Federal Information Processing Standards Publication 180-2: SECURE HASH STANDARD.
Date: August 1, 2002.
http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

[FIPS198-02] Federal Information Processing Standards Publication 198: The Keyed-Hash Message Authentication Code (HMAC).
Date: March 6, 2002.
http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf

[HKN05] Shai Halevi, Paul A. Karger and Dalit Naor. Enforcing confinement in distributed storage and a cryptographic model for access control, 2005. Cryptology ePrint Archive: Report 2005/169.

[ACF+02] Alain Azagury, Ran Canetti, Michael Factor, Shai Halevi, Ealan Henis, Dalit Naor, Rinetzky Noam, Ohad Rodeh, and Julian Satran. A two layered approach for securing an object store network. In Proceedings of the First International IEEE Security in Storage Workshop, pages 10–23, Greenbelt, MD, 11 December 2002.