Capability based Command Security

SCSI commands standard proposal

IBM Research Lab in Haifa

March 2007

Comment: Sivan: George, can you please make the appropriate T10 proposal header. I think the table of content is good to have although it's not typically included in proposals. This one is long enough to deserve TOC.

Table of Contents

Changes to SPC-4

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

1 Definitions, symbols, abbreviations, and conventions

(These are additions to section 3 of SPC-4.)

1.1 Definitions

(These are additions to section 3.1 of SPC-4.)

[Sivan: This section is almost a complete rewrite from previous version]

1.1.1 CbCS Capability

A data structure that specifies defined access to a logical unit for specific commands. See 1.6.1.

1.1.2 CbCS capability key

The integrity check value (see 1.1.11) in a CbCS credential (see 1.1.3) returned by the CbCS management device server (see 1.1.5) in response to the RECEIVE CREDENTIAL command (see 10). See 1.3.2.

1.1.3 CbCS Credential

A data structure containing a capability that is protected by an integrity check value (see 1.1.4) that is sent to and used by an application client in order to grant defined access to a logical unit for specific command functions (see **Error! Reference source not found.**). See 1.8.

1.1.4 CbCS management application client

An application client that manages secret keys (see 1.1.12) stored in logical units. See 1.4.

1.1.5 CbCS management device server

A device server that prepares CbCS credentials (see 1.1.3) that contain CbCS capabilities (see 1.1.1) in response to application client requests. See 1.3.

1.1.6 CbCS master key

A set of secret keys (see 1.1.12) that consist of a CbCS master authentication key (see 1.1.7) and a CbCS master generation key (see 1.1.8).

1.1.7 CbCS master authentication key

A secret key (see 1.1.12) used to generate integrity check values (see 1.1.11) for CbCS credentials (see 1.1.3). See 2.2.

1.1.8 CbCS master generation key

A secret key (see 1.1.12) used to generate new CbCS working keys (see 1.1.10).

1.1.9 CbCS validation tag

Allows the device server to validate a credential and determine if the capability has been tampered with. See 1.3.2.

1.1.10 CbCS working key

A secret key (see 1.1.12) that is used in generating integrity check values (see 1.1.11) for CbCS credentials (see 1.1.3). See 2.2.

1.1.11 Integrity check value

A value computed using a security algorithm (e.g., HMAC-SHA1), a secret key (see 1.1.12), and an identified set of data that protects the integrity of that identified set of data. See 1.8.

1.1.12 Secret key

A cryptographically generated value that is known only to a limited set of authorized entities. See 2.2.

1.1.13 Security token

A value representing an I_T nexus (see 3.1.17) known to both the application client and device server. (GP – Still looking to eliminate this are replace it with a I_T nexus ID)

3.2 Acronyms

CbCS Capability based Command Security

2 Capability based Command Security

1.2 Overview

CbCS is a credential-based access control system. The security model is composed of the following components:

a) A SCSI target device;

b) A security manager consisting of:

   a. A CbCS management device server

   b. A CbCS management application client

c) Application clients.

(Editors note: This needs a UML class diagram to accurately describe the interaction of the security classes to the classes we already have. The UML should also include the CbCS management application client and a CbCS management device server.)

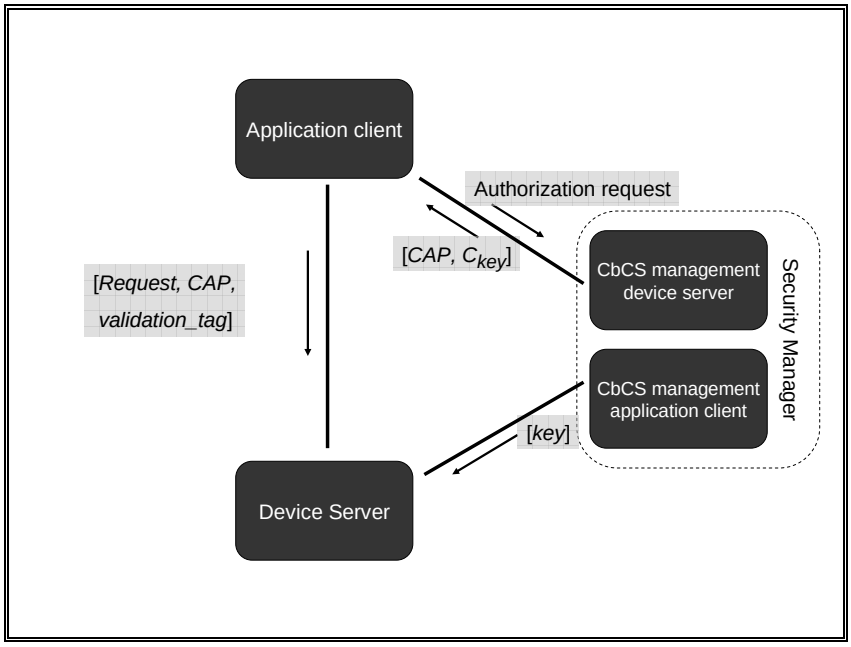
Comment: George, can you take a first crack at that UML? You should know better how you want it to look…

The CbCS management device server manages access control policy. CbCS Credentials are prepared by the CbCS management device server based on that access control policy.

Controlling access to a logical unit requires coordination between secret keys and security attributes set by the CbCS management application client and credentials generated by the CbCS management device server. The mechanism for coordination between the CbCS management device server and the CbCS management application client is not defined in this standard.

Figure 1 shows the flow of transactions between the components of a CbCS capable SCSI domain.

Figure 1 - The CbCS security model



1.3 CbCS management device server

1.3.1 CbCS management device server overview

The application client requests capabilities and CbCS capability keys from the CbCS management device server. A CbCS management device server returns a CbCS capability key (C_{key}) with each CbCS credential giving the application client access to a specific logical unit. The application client sends the CbCS capability to that logical unit's device server as part of a CbCS encapsulated command which allows the device server to authenticate the CbCS capability with an integrity check value (see 1.8.2, 2.1.1).

The CbCS management device server shall authenticate the application client unless NOSEC security method is used (see 1.7.3), but secure access between the application client and device server is provided without requiring authentication of the application client by the device server. It is sufficient for the device server to verify the CbCS capabilities and integrity check values sent by the application client.

1.3.2 CbCS Credentials

The RECEIVE CREDENTIAL command is used to request an CbCS credential from the CbCS management device server (see 10).

If the CAPKEY security method is used, the device server shall validate each command received from an application client to confirm that:

a) The credential has not been tampered with (i.e., that the CbCS credential was generated by the CbCS management device server and includes an integrity

check value generated using a secret key known only to the CbCS management device server, the CbCS management application client and the device server);

b) The credential was obtained by the application client from the CbCS management device server or through delegation by another application client (i.e., that the application client knows the capability key that is associated with the credential and has used the CbCS capability key to provide a proper integrity check value for the command); and

c) The requested SCSI command encapsulated in the received CDB is permitted by the capability in the credential (see 4.1).

If the NOSEC security method is used, the device server shall validate each command received from an application client to confirm that the requested SCSI command encapsulated in the received CDB is permitted by the CbCS capability in the CbCS credential (see 4.1).

A CbCS credential includes a CbCS capability and an integrity check value. The integrity check value in a CbCS credential returned by the CbCS management device server in response to the RECEIVE CREDENTIAL command (see 10) is used as the secret key to generate the integrity check value in a CbCS credential sent by the application client in a CbCS encapsulation (see 6).

The integrity check value in a CbCS credential returned by the CbCS management device server in response to the RECEIVE CREDENTIAL command is the CbCS capability key. The integrity check value in a CbCS credential sent by the application client in a CbCS encapsulation is the validation tag.

The validation tag allows the device server to validate an CbCS credential and determine if the CbCS capability has been tampered with (e.g., an application client that has just the CbCS capability but not the CbCS capability key is unable to generate CbCS credentials with a valid validation tag in the integrity check value). Delegation of a CbCS credential is permitted, if an application client delegates both the CbCS capability and the CbCS capability key.

1.4 CbCS management application client

The CbCS capability keys are computed using secret keys that are shared between the CbCS management device server and the device server. The secret keys are managed by the CbCS management application client in conjunction with the CbCS management SCSI initiator device. The command integrity check values (i.e. the validation tags) are computed using CbCS capability keys. This standard includes SCSI commands for the CbCS management application client to set and manage the secret keys stored in the logical unit or a well known logical unit (see 8.1, 9.1).

If the secret key is stored in a well known logical unit then the key is shared between all logical units within the target device but is only used by a logical unit if there has been no secret key assigned to that logical unit (i.e., a secret key assigned to a logical unit always overrides any secret key assigned to a well known logical unit).

1.5 Trust assumptions

After the logical unit is a trusted (i.e., after an application client authenticates that it is communicating with a specific logical unit), the application client trusts the device server to:

a) Provide integrity for stored data;

b) Perform the security protocol and functions defined for it by this standard; and

c) Not be controlled in a way that operates to the detriment of the application client's interests. (GP – This item should be deleted as it basicly states the logical unit should not do bad things)

The CbCS management device server and the CbCS management application client are trusted after:

a) the CbCS management device server is authenticated by the application client; and

b) the CbCS management application client is authenticated by the logical unit.

The CbCS management device server and the CbCS management application client are trusted to:

a) Securely store long-lived secret keys;

b) Grant credentials to application clients according to access control policies that are outside the scope of this standard; and

c) Perform the defined security functions.

The application client is not trusted. However, the CbCS security model is defined so that the application client receives service from the device server only if it interacts with both the CbCS management device server and the device server in ways that assure the propriety of the application client's actions.

CbCS management application clients and CbCS management device servers are trusted to protect CbCS capability keys from disclosure to unauthorized entities.

Use of the capability expiration time requires synchronization between the clocks of the CbCS management device server and the device server.

Communications between the CbCS management application clients, application clients, CbCS management device servers, and device servers are trusted based on the requirements shown in Table 1.

Table 1 - Communications trust requirement

Connection	Communication trust requirement
application client <--> device server	message integrity [a]
application client <--> CbCS management device server	Confidentiality and integrity
CbCS management application client <--> device server	message integrity
CbCS management device server <--> CbCS management application client	Confidentiality and integrity
Confidential communications are protected from eavesdropping by methods outside the scope of this standard. Message integrity assures that the message received is the one that was sent (i.e., no tampering occurred). Messages in which tampering is detected shall be discarded. [a] Message integrity is sufficient for security of the CbCS access control. Confidentiality of the	

data transferred between the application client and the device server may be required to prevent passive network attacks and implemented by other means.

1.6 Policy Management

Policy management shall be performed by the CbCS management application client and the CbCS management device server as follows:

a) The CbCS management device server provides access policy controls to application clients using policy-coordinated CbCS capabilities; and

b) The CbCS management application client, in concert with the CbCS management device server and the device server, prevents unsecured access to a logical unit.

The policy management is confined to the CbCS management application client and CbCS management device server. The communication of policy management information may occur in a manner outside the scope of this standard.

1.6.1 CbCS Capabilities

1.6.1.1 Overview

All CbCS encapsulations contain a CbCS capability descriptor that specifies the command functions (e.g., read, write, attributes setting, attributes retrieval) that the device server is allowed to process in response to the encapsulated SCSI CDB.

The device server shall validate that the requested functions are allowed by the CbCS capability based on:

a) The type of functions; and

b) The logical unit.

The policies that determine which CbCS capabilities are provided to which application clients are outside the scope of this standard.

The CbCS management device server shall deliver CbCS capabilities to application clients as follows:

a) If the security method in use for the logical unit is NOSEC (see 1.7.3), then the CbCS management device server may:

   A) Allow application clients to prepare their own CbCS capabilities; or

   B) Coordinate the preparation of CbCS capabilities for multiple application clients in response to requests;

   or

b) If a security method in use for the logical unit is CAPKEY (see 1.7.4), then the CbCS management device server shall prepare of CbCS capabilities by:

   A) Requiring application clients to request CbCS credentials and CbCS capabilities; and

   B) Preparing CbCS capabilities only in response to application client requests.

A CbCS capability descriptor is included in an CbCS credential returned by the CbCS management device server in response to the RECEIVE CREDENTIAL command (see

10), and in CbCS encapsulation parameters (see 6) to enable the device server to verify that the application client is allowed to perform the command functions requested by the encapsulated CDB. The capability format is defined in 10.1.1.

The CbCS capability descriptor specifies:

a) the security method to apply in validating the CbCS credential;

b) the cryptographic parameters used in generating the CbCS credential integrity check value;

c) an expiration time of the CbCS capability;

d) a permission bit mask that specifies which command functions are authorized by the CbCS capability;

e) a logical unit descriptor which uniquely identifies the logical unit to which the CbCS capability pertains; and

f) a policy access tag which is used for CbCS credential revocation.

Effective use of the CbCS capability expiration time requires synchronization between the clocks of the CbCS management device server and the device server. The method for synchronizing the clocks is outside the scope of this standard. Use of the CbCS capability expiration time is optional – a value of zero indicates that the capability has no expiration time. (GP – The information about the zero value needs to be in the field description not here)

The CbCS capability includes:

a) an audit function, that may be used in a vendor specific manner to limit the delegation or prevent leakage of the CbCS capability to other application clients; and

b) a designation descriptor (see spc4r09) that uniquely identifies a logical unit. If the CbCS capability applies to a well-known logical unit, the designation descriptor applies to the target device in which it resides.

In order to implement CbCS credential based on logical unit unique identifier, the same identifier type shall be:

a) used in the access control policies set in the policy manager;

b) returned by the device server in Device Identification VPD page (Inquiry page 83h);

c) used by the application client to identify the device and request the corresponding CbCS credential from the CbCS management device server; and

d) returned by the CbCS management device server in response in the CbCS capability in response to the application client's CbCS credential request.

A CbCS management application client may block CbCS capability-based access to a logical unit by:

a) changing the policy access tag attribute associated with a logical unit (see 9.1.2);

or

b) changing or invalidating the secret keys shared with the device server.

1.6.1.2 CbCS Capability validation

The device server shall validate the CbCS capability descriptor included in the CbCS encapsulation (see 6) as follows:

a) Verify that the CAPABILITY FORMAT field value is set to 1h. If the CAPABILITY FORMAT field value is other than 1h, 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.

b) If the CAPABILITY EXPIRATION TIME field contains a non-zero value, then compare the CAPABILITY EXPIRATION TIME field to the current time (i.e., the current number of millisecond passed since midnight, 1 January 1970 UT). If the CAPABILITY EXPIRATION TIME field value is smaller than the current time value, 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.

c) Verify that the designation descriptor matches the addressed logical unit, or the addressed target device in case a well-known logical unit is addressed. If they don't match, 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.

d) If any of the LBA START field or LENGTH field contains nonzero value, verify that the LBA range affected by the encapsulated command is fully covered by the LBA range specified by the LBA START field and the LENGTH field (i.e., the affected LBA range is within the range spanning from [LBA START] to [LBA START + LENGTH - 1]). (GP – CbCS should be limited to logical units not LBA ranges of logical units)

e) If the POLICY ACCESS TAG field in the CbCS capability descriptor contains a non-zero value, then compare the POLICY ACCESS TAG field to the Policy Access Tag of this logical unit. If they don't match, 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.

f) Verify that the encapsulated command is permitted by the PERMISSIONS BIT MASK field in the CbCS capability descriptor in the CDB. If the requested command is not permitted, 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.

1.7 Security Methods

1.7.1 Overview

This standard defines two security methods (see Table 2).

Table 2 - CbCS security methods

Method	Description	Without a secure channel	With a secure channel

Method	Description	Without a secure channel	With a secure channel
NOSEC	No security [a]	No security	Network-level integrity
CAPKEY	Access control [b]	Verification of credentials, vulnerable to some network attacks	Protection from network attacks

[a] The device server verifies the CbCS capability allows the operation but does not verify the authenticity of the CbCS capability prior to processing a command.

[b] Access Control Security is based on the protocol presented and analyzed in [ACF+02]. CAPKEY verifies that the application client rightfully obtained the credential it is presenting.

1.7.2 General

If the device server receives a command for a logical unit:

a) that has CbCS enabled:

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

c) the encapsulation type is set to 10h (i.e. CbCS encapsulation),

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

If the device server receives a command for a logical unit:

a) that has CbCS enabled:

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

c) the received command requires authorization as described in 4.1,

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.

If the device server receives a command for a logical unit:

a) that has CbCS enabled;

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

c) the encapsulations do not include an encapsulation type set to 10h (i.e. CbCS encapsulation); and

d) the encapsulated command requires authorization as described in 4.1,

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.

The validations performed on a command using the NOSEC method are a subset of the validations performed on a command using the CAPKEY method. Therefore, 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. (GP – I don't think this paragraph has any useful information a therefore should be deleted.)

1.7.3 The NOSEC security method

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

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

Preparing CbCS credentials for the NOSEC security method does not require the knowledge of any secret keys and may be done by the application client without coordination with the CbCS management device server. The CbCS capability descriptor (see 1.6.1, 10.1.1) is set with the SECURITY METHOD field set to NOSEC. The integrity check value is set to zero.

The device server validates the CbCS capability as described in 1.6.1.2.

1.7.4 The CAPKEY security method

1.7.4.1 Overview

The CAPKEY security method validates the integrity of the CbCS capability information in each command. It provides security when the service delivery subsystem between the device server and application client is secured.

The integrity of the CbCS capability shall be validated before any other command processing is done, including CbCS capability validation.

Given a CbCS credential and a channel, the protocol ties the CbCS credential to the channel via a validation tag. The validation tag is computed by the client as $F_{C_{key}}^{pr}(SecurityToken)$,

Where:

SecurityToken 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.

Comment: Sivan: OSD says the security token shall be random. Why?

(GP – A channelID looks like an I_T nexus ID to me. I would rather define a new identifier rather the an new name for something that already exists)

C_{key} is the CbCS capability key associated with the command (see 1.7.4.2).

SecurityToken is chosen by the device server. An application client may request the value of the *SecurityToken* (see 8.1.5). The device server compares the channel on which a request was received and its *SecurityToken*, and verifies that the validation tag attached to the request equals $F_{C_{key}}^{pr}(SecurityToken)$.

To ensure the request is authenticated by the application client who obtained the CbCS credential, the CbCS capability key (C_{key}) with which the validation tag $F_{C_{key}}^{pr}(SecurityToken)$ is computed depends on the CbCS capability.

CbCS capabilities and integrity check values may be reused as follows:

a) The application client may reuse the CbCS capability and C_{key} on multiple commands for the same logical unit(s);

b) The application client is required to calculate the validation tag once per I_T_L nexus;

c) The C_{key} and validation tag shall be calculated the first time the device server receives a given CbCS capability on a given I_T_L nexus and may be reused in processing every command received on the I_T_L nexus; and

1.7.4.2 Computing the CbCS capability key

When preparing CbCS credentials (see 1.8.1) and validating CbCS credentials (see 2.1.1), the CbCS capability key shall be computed by the CbCS management device server and the device server using:

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

b) If the value of the KEY VERSION field in the CbCS capability descriptor is nonzero, the secret key to by used is specified by the KEY VERSION field, otherwise the authentication master key is used as the secret key; and

c) The CbCS capability descriptor as the input data.

The CbCS capability key is placed in the INTEGRITY CHECK VALUE field of the CbCS credential returned in the RECEIVE CREDENTIAL command.

1.7.4.3 Computing the validation tag

When preparing CbCS credentials (see 1.8.2), the validation tag shall be computed by the application client using:

1) The algorithm specified in the INTEGRITY CHECK VALUE ALGORITHM field in the CbCS capability descriptor;

2) The CbCS capability key returned from the RECEIVE CREDENTIAL command in the INTEGRITY CHECK VALUE field;

3) The security token returned by the device server in the CbCS Attributes page (see 8.1.5) as the input data.

The validation tag is placed in the INTEGRITY CHECK VALUE field of the CbCS credential passed by the application client in the CbCS encapsulation (see 6).

1.8 CbCS Credentials

A CbCS credential authorizes specific access to a specific logical unit. It consists of a CbCS capability and an integrity check value. The CbCS capability descriptor (see 1.6.1) identifies the logical unit and specifies the specific access rights and parameters specifying how it shall be validated by the device server. The integrity check value authenticates the CbCS capability and is used for validation.

There are two types of CbCS credentials used in the CbCS protocol:

a) A CbCS credential is transferred from the CbCS management device server to an application client over a communications mechanism that meets the requirements specified in 1.5 with the CbCS credential being returned in response to the RECEIVE CREDENTIAL command (see 10); and

b) A CbCS credential is transferred from the application client to the device server over a communications mechanism that meets the requirements specified in 1.5 with the credential being placed in the encapsulation parameters of the CDB CbCS encapsulation (see 6).

1.8.1 Preparing CbCS credentials by the CbCS management device server

In response to a request from an application client, the CbCS management device server shall prepare and return a CbCS credential (see Table 19) as follows:

1) If the access controls policy does not authorize the application client's request, no CbCS credential shall be returned to the requesting application client, (i.e. the CRED PRSNT bit (see xx.xx.xx) in the returned parameter data shall be set to zero;

2) Prepare the CbCS capability and insert it in the CbCS credential as follows:

   a. Setting the SECURITY METHOD field to the value of the corresponding field in the Attributes CbCS page (see 8.1.5) of the logical unit for which the credential is requested;

   b. Setting the KEY VERSION field to the number of the working key secret key used to compute the credential integrity check value; (GP – what is a "working key secret key"?)

   c. Setting the INTEGRITY CHECK VALUE ALGORITHM field to the value that specifies the algorithm used to compute all integrity check values related to this CbCS credential. The algorithm shall be one of those identified by the supported integrity check value algorithm attributes in the CbCS capabilities page (see 8.1.4) of the logical unit for which the credential is requested;

   d. Setting the CAPABILITY EXPIRATION TIME field to a value that is consistent with the CbCS policy;

   e. May set the AUDIT field in a vendor specific manner;

   f. Setting the PERMISSIONS BIT MASK descriptor to a value that is consistent with the policy;

   g. Setting the POLICY ACCESS TAG field to a value that matches the POLICY ACCESS TAG attribute in the Attributes CbCS page (see 8.1.5) of the logical unit for which the CbCS credential is requested. The value zero may be used to prevent revocation by changing the policy access tag attribute of the logical unit;

   h. Setting the LU DESCRIPTOR TYPE field, LU DESCRIPTOR LENGTH field, and LU DESCRIPTOR field to those of the logical unit to which the credential is requested; and

   i. If the security method in use is CAPKEY, then compute the CbCS capability key as described in 1.7.4.2, and place it in the INTEGRITY CHECK VALUE field in the CbCS credential. If the security method in use is NOSEC, set the INTEGRITY CHECK VALUE field to zero;

   and

3) Return the CbCS credential to the application client with the integrity check value serving as the CbCS capability key.

Use of the CbCS capability expiration time (see item d in step 2) requires synchronization between the clocks of the device server, the CbCS management

application client, and the CbCS management device server. The protocol for synchronizing the clocks is not specified in this standard, however, the protocol should be implemented in a secure manner (e.g., it should not be possible for an adversary to set the clock in the device server backwards to enable the reuse of expired CbCS credentials). The REPORT TIMESTAMP command and SET TIMESTAMP command encapsulated with CbCS encapsulation may be used by the CbCS management application client for this purpose.

Security management commands issued by the CbCS management application client to the device server require that the integrity check value is computed using the authentication CbCS master key rather than a CbCS working key. The list of commands requiring use of the CbCS master key is in 4.1.1. If the CbCS master key is used to compute the CbCS credential integrity check value then the KEY VERSION field in the CbCS capability descriptor shall be set to zero. (GP – there is no entry in the glossary for CbCS working key. This needs to be fixed.)

For CbCS credentials returned by the CbCS management device server in response to the RECEIVE CREDENTIAL command (see 10), only CbCS working keys shall be used in computing the INTEGRITY CHECK VALUE field.

1.8.2 Preparing CbCS credentials by the application client

The client shall prepare the CbCS credential for sending it to the device server in the CDB CbCS encapsulation parameters as follows:

a) If the CAPKEY security method is enabled, copy the CbCS capability descriptor received from the CbCS management device server in response to the RECEIVE CREDENTIAL command into the CbCS capability descriptor parameter of the CDB CbCS encapsulation. If the NOSEC security method enabled, prepare the CbCS capability descriptor as described in 1.8.1; and

b) If the CAPKEY security method is enabled, compute the validation tag as described in 1.7.4.3 and place it in the INTEGRITY CHECK VALUE parameter of the CDB CbCS encapsulation.

2.1.1 Validating CbCS credentials by the device server

The device server shall validate CbCS credentials as follows:

a) If the CAPKEY security method is enabled and the SECURITY METHOD field in the CbCS capability descriptor is other than CAPKEY, 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; and

b) If the CAPKEY security method is enabled, then:

a. Compute the CbCS capability key as described in 1.7.4.2;

b. Compute the validation tag as described in 1.7.4.3; and

c. Compare the computed validation tag with the INTEGRITY CHECK VALUE field in the CbCS encapsulation parameters. 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;

and

c) Verify the CbCS capability descriptor as described in 1.6.1.2.

2.2 Secret Keys

1.8.3 Overview

All CbCS credentials are based on a secret key that is shared between the device server, the CbCS management application client that manages its security attributes, and the CbCS management device server that grants CbCS credentials to application clients. Keys shall be refreshed regularly.

Secret key management requirements are as follows:

2 The CbCS management application client should replace the logical unit's secret keys in a secure manner even if the channel it has with the logical unit is not secure;

3 The device server shall support multiple CbCS working keys; and

4 The CbCS management application client shall contain a random source for generating secret keys.

Each logical unit has a CbCS master key and a set of CbCS working keys assigned to it. A single set of secret keys may be shared among multiple logical units within a target device.

The CbCS working keys are used to generate the CbCS capability keys that are used by application clients to access logical units. CbCS working keys should be refreshed frequently (e.g., hourly). A secret key refresh shall invalidate all CbCS credentials generated by that secret key. The device server shall support up to 16 refreshed versions of the secret key as valid (i.e., define multiple secret keys that are concurrently valid). The secret key version field in the CbCS capability is used to indicate which secret key shall be used in the validation process (see 10.1.1).

When setting a new secret key, the CbCS management application client assigns the secret key with a version number. The KEY VERSION field in the CbCS capability descriptor is set to the version number of the CbCS working key used in computing the CbCS capability key (see 10.1.1). The device server uses the KEY VERSION field to determine which secret key to use in validating an CbCS credential in a CbCS encapsulated command (see 2.1.1).

A CbCS master generation key is used to generate working secret keys. A CbCS master authentication key is used to generate CbCS credentials for commands to set and refresh keys, and modify device security attributes. The CbCS master key (i.e., the pair of CbCS master generation key and CbCS master authentication key) may be refreshed. Refreshing the CbCS master key is accomplished by a Diffie-Hellman key exchange algorithm that ensures forward secrecy of the CbCS master key. This algorithm is carried over a sequence of commands as follows:

1) SECURITY PROTOCOL OUT command specifying the CbCS protocol and the Set Master Key, Seed Exchange page (9.1.4);

2) SECURITY PROTOCOL IN command specifying the CbCS protocol and the Set Master Key, Seed Exchange page (8.1.6); and

3) SECURITY PROTOCOL OUT command specifying the CbCS protocol and the Set Master Key, Change Master Key page (9.1.5).

Separate sets of the CbCS master key and the CbCS working keys may be used for each logical unit, or a single set may be used for all logical units served by a security protocol well-known logical unit's device server as follows:

1) A single set of CbCS master keys and CbCS working keys is used by the device server through the SECURITY PROTOCOL well-known logical unit. This set of secret keys serves all the logical units within the SCSI target device;
2) A separate set of CbCS master keys and CbCS working keys is used for each logical unit within the SCSI target device; and
3) A single set of CbCS master keys and CbCS working keys is used by the device server through the SECURITY PROTOCOL well-known logical unit. In addition, a separate set of CbCS master key and CbCS working keys may be used for any logical unit within the SCSI target device. The single set of secret keys serves any logical unit that does not have its own set of keys.

Comment: Sivan: Removed the sentence that says the CbCS management application client may use the same set of keys for a set of LUs. Since it's based on DH key exchange, I don't think it is possible to generate identical keys.

4.1.1 Secret key usage in commands

Every CbCS credential prepared by the CbCS management device server includes an integrity check value field containing a CbCS capability key that is computed using either a CbCS working key or the CbCS master authentication key associated with the logical unit.

The CbCS authentication master key shall be used in preparing CbCS credentials for commands that set secret keys. For other commands, the secret key that shall be used is one of the CbCS working keys associated with the logical unit. The KEY VERSION field in the CbCS capability descriptor shall be set to a value that identifies that particular working key (see 10.1.1).

The following commands involve setting secret keys and require using the CbCS master authentication key for preparing the credential:

a) SECURITY PROTOCOL OUT command specifying the CbCS security protocol and the Set Key page (see 9.1.3);

b) SECURITY PROTOCOL OUT command specifying the CbCS security protocol and the Set Master Key, Seed Exchange page (see 9.1.4);

c) SECURITY PROTOCOL IN command specifying the CbCS security protocol and the Set Master Key, Seed Exchange page (see 8.1.6); and

d) SECURITY PROTOCOL OUT command specifying the CbCS security protocol and the Set Master Key, Change master Key page (see 9.1.5).

[Editor's note (Sivan): Do we want to specify the master key for credential for SECURITY PROTOCOL OUT command specifying the CbCS security protocol and the Set attributes CbCS page (see 9.1.2)? I think not…]

If the CbCS encapsulated command is SECURITY PROTOCOL OUT command specifying the CbCS security protocol and the Set Master Key, Change Master Key page, then the secret key that shall be used is the next CbCS 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.

3 Computing updated CbCS master generation keys and new CbCS authentication keys

When processing the commands that involve setting secret keys (i.e., the Set Key CbCS page (see 9.1.3) and the Set Master Key CbCS pages (see 9.1.4, 8.1.6, and 9.1.5)), the device server shall compute new CbCS master generation keys and CbCS working keys as follows:

a) The algorithm specified in the INTEGRITY CHECK VALUE ALGORITHM field in the CbCS capability descriptor; [Editor's note: Should we define a separate field encoding this algorithm rather than using the one used in the credential?]

b) The input secret key value shall be one of the following:

   a) For a Set Key CbCS page (see 9.1.3), the CbCS master generation key; or

   b) For a CbCS master key computed following the processing of the Set Master Key, Change Master Key CbCS page (see 9.1.5), the previous CbCS master generation key shall be used;

and

c) The input data (i.e., seed) shall be one of the following:

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

   b) For a Set Master Key, Change Master Key CbCS page, the value computed after successful completion of the SECURITY PROTOCOL IN command specifying the Set Master Key, Seed Exchange CbCS page (see 8.1.6) and updated by the Set Master Key, Change Master Key CbCS page (see 9.1.5).

When processing the commands for setting the CbCS master key (i.e., the Set Master Key CbCS pages (see 9.1.4, 8.1.6, and 9.1.5)), the device server shall compute the new CbCS master authentication key as follows:

a) The algorithm specified in the INTEGRITY CHECK VALUE ALGORITHM field in the CbCS capability descriptor; [Editor's note: Should we define a separate field encoding this algorithm rather than using the one used in the credential?]

b) The input key value shall be the CbCS master generation key prior to processing the command; and

c) The input data shall be the seed value computed (see 8.1.6) with the least significant bit changed as follows:

   a) If the seed value least significant bit is zero, then the least significant bit shall be set to one; or

   b) If the seed value least significant bit is one, then the least significant bit shall be set to zero.

4.1 CbCS interactions with commands and task management functions

4.1.1 Association between commands and command functions

The PERMISSIONS BIT MASK descriptor in the CbCS capability (see 1.6.1) specifies which command functions are allowed by this CbCS capability. When processing CbCS encapsulation 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 processing the request specified by the CbCS encapsulated SCSI command.

The associations between commands and command functions as specified in this subclause for commands defined in this standard. Other SCSI command set standards may specify associations between commands and command functions pertaining to the specific device type.

If the device server receives a command that requires CbCS encapsulation (see 6) according to Table 3 or any device specific command set standard pertaining to the received command, and the command is not encapsulated with CbCS encapsulation, then 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 device server receives a command that does not require CbCS encapsulation (see 6) according to Table 3 or any device specific command set standard pertaining to the received command, and the command is encapsulated with CbCS encapsulation, then the command may be processed by the device server.

Table 3 - Associations between commands and command functions

Command	Permissions (x.x.x) (GP – make this change elsewhere)						Not allowed (a)
	DATA READ	DATA WRITE	ATTR READ	ATTR WRITE	SEC MGMT	RESRV	
ACCESS CONTROL IN							v
ACCESS CONTROL OUT							v
CHANGE ALIASES							
EXTENDED COPY							v
INQUIRY							
LOG SELECT			v				
LOG SENSE				v			
MANAGEMENT PROTOCOL IN							
MANAGEMENT PROTOCOL OUT							
MODE SELECT(6)				v			
MODE SELECT(10)				v			
MODE SENSE(6)			v				
MODE SENSE(10)			v				
PERFORM TASK MANAGEMENT FUNCTION					v		
PERSISTENT RESERVE IN			v				
PERSISTENT RESERVE OUT						v	
READ ATTRIBUTE			v				
READ BUFFER	TBD	TBD	TBD	TBD	TBD	TBD	TBD

READ MEDIA SERIAL NUMBER			v				
RECEIVE COPY RESULTS							v
RECEIVE CREDENTIAL							
RECEIVE DIAGNOSTIC RESULTS			v				
REPORT ALIASES			v				
REPORT IDENTIFYING INFORMATION			v				
REPORT LUNS							
REPORT PRIORITY			v				
REPORT SUPPORTED OPERATION CODES							
REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS							
REPORT TARGET PORT GROUPS							
REPORT TIMESTAMP			v				
REQUEST SENSE			v				
SECURITY PROTOCOL IN	[Editor's note: TBD. We have to define associations per security protocol. For IKEv2-SCSI no encapsulation shall be required otherwise we get chicken-n-egg problem]						
SECURITY PROTOCOL OUT							
SEND DIAGNOSTIC				v			
SET IDENTIFYING INFORMATION				v			
SET PRIORITY				v			
SET TARGET PORT GROUPS							
SET TIMESTAMP				v	v		
TEST UNIT READY							
WRITE ATTRIBUTE				v			
WRITE BUFFER	TBD	TBD	TBD	TBD	TBD	TBD	TBD
[a] A check mark in this column means that the command shall not be supported for a logical unit that has the CbCS bit set in the extended inquiry data							

4.1.2 Task management functions

If the CbCS bit is set to one in the extended inquiry data, all SAM-4 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 11) allows SAM-4 task management functions to be processed under the protection of CbCS.

Comment: Sivan: What's the impact of this on existing host systems?!

[Editor's note (Sivan): CbCS is per LU. Do Task management functions pertain to LU? If not, do we have an issue here with having both CbCS and non-CbCS LUs in the same device?]

(GP- This is a very nasty requirement.)

4.2 Security attributes

Device servers supporting CbCS may support one or more of the following security attributes:

a) SCSI target device based;

b) logical unit based;

c) changeable; or

d) non-changeable.

Device servers may support the following security attributes:

a) only SCSI target device based;

b) only logical unit based; or

c) both.

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

SCSI target device attributes are queried and modified through the SECURITY PROTOCOL well-known logical unit (see 8.5 [spc4r09])

Table 4 specifies the CbCS attributes.

Table 4 - CbCS attributes

Security attribute name	Length (bytes)	SCSI target device or logical unit specific	Application client settable
Supported security methods	n*2	**SCSI t**arget device	No
Default security method	2	**SCSI t**arget device	Yes
Security method	2	**SCSI t**arget device/Logical unit	Yes
Supported integrity check value algorithms	n*2	**SCSI t**arget device	No
Supported DH groups	n*2	**SCSI t**arget device	No
Clock	6	**SCSI t**arget device	Yes
CbCS master key identifier	8	**SCSI t**arget device / Logical unit	No [d]
CbCS working key identifier	16*8	**SCSI t**arget device / Logical unit	No [d]
LU initial policy access tag	4	**SCSI t**arget 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 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 nexus

[d] The secret key identifier is set by the application client when a new secret key is generated as described in 9.1.3 and 9.1.5. It is not settable by means of setting CbCS security attributes described in 9.1.2

Comment: How does the application client know how much space to allocate if this is undefined?

The supported security methods attribute is used by the device server to report its supported security methods. See 1.7.

The default security method attribute is the security method the device server shall apply to a newly created CbCS logical unit, if a security method is not specified at logical unit creation time. The default security method shall be one of the supported security methods.

Comment: Is creation of logical units defined in any standard?

The supported integrity check value algorithms security attribute is used by a device server to report its supported integrity check value algorithms. Integrity check value algorithms shall be used to compute integrity check values. See [06-449r2 - 7.7.4.11.1.3 Integrity Algorithm (INTEG) identifiers].

The supported DH group security attribute is used by a device server to report the DH groups it supports for the Diffie-Hellman key exchange with the application client that is processed as part of setting a new CbCS master key (see 9.1.4). See [06-449r2 - 7.7.4.11.1.4 Diffie-Hellman Group (D-H) identifiers].

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 FFFF FFFFh (see **Error! Reference source not found.**).

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 (see **Error! Reference source not found.**).

Setting and querying security attributes are used by the application client by issuing the SECURITY PROTOCOL IN command and SECURITY PROTOCOL OUT command with the CbCS security protocol (see 8.1, 9.1).

5 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	CbCS	98	0	
xxh - FFh	Reserved		0	

6 CbCS encapsulation

[Editor's note: I'm not sure where exactly this section should be added in SPC-4.]

The CbCS encapsulation allows the application of security to a SCSI command using the parameters specified in this subclause.

Support for CbCS encapsulation type is mandatory if the CbCS bit in extended INQUIRY data (see 7) is set to one.

The encapsulated CDB may be any any CDB defined in any SCSI standard.

Comment: I wouldn't add text to rule it out, but does it make any sense to use CbCS w/ OSD?

Table 5 - CbCS encapsulation descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
0	NEXT ENCAPSULATION TYPE							
1	Reserved							
2	CbCS capability descriptor							
81								
82	INTEGRITY CHECK VALUE							
145								

The CbCS capability descriptor is defined in 10.1.1.

The CbCS capability descriptor and the INTEGRITY CHECK VALUE field shall be prepared by the application client as described in 1.8.2.

The device server shall validate the CbCS capability descriptor and the INTEGRITY CHECK VALUE field as described in 2.1.1.

7 Extended INQUIRY Data VPD page

[Change in 7.6.4 Extended INQUIRY Data VPD page]

The Extended INQUIRY Data VPD page (see table 361) provides the application client with a means to obtain information about the logical unit.

Table 361 — Extended INQUIRY Data VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (86h)							
2	Reserved							
3	PAGE LENGTH (3Ch)							
4	Reserved		SPT			GRD_CHK	APP_CHK	REF_CHK

Bit Byte	7	6	5	4	3	2	1	0
5	Reserved			GROUP_ SUP	PRIOR_S UP	HEADSU P	ORDSUP	SIMPSUP
6	Reserved					COR_D_S UP	NV_SUP	V_SUP
7	Reserved							LUICLR
8	Reserved							CbCS
9 63	Reserved							

<Unchanged text here>

A Capability based Command Security (CbCS) bit set to one indicates that the logical unit has CbCS (see 2). A CbCS bit set to zero indicates that the logical unit does not support CbCS.

(GP – The information in this section below this note belongs in the model not in the command description)

If the CbCS bit is set to one, the target device shall support the following commands and parameters:

a) Encapsulated SCSI command (see 4.3.4.2 [07-029r1]);

b) CbCS encapsulation type (see 6);

c) SECURITY PROTOCOL IN specifying the CbCS security protocol (see 8.1).

d) SECURITY PROTOCOL OUT specifying the CbCS security protocol (see 9.1).

8 Changes in SECURITY PROTOCOL IN command

[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	CbCS	6.29.3
08h - 1Fh	Reserved	
20h	Tape Data Encryption	SSC-3
21h - EDh	Reserved	
EEh	Authentication in Host Attachments of Transient Storage Devices	IEEE P1667

Code	Description	Reference
EFh	ATA Device Server Password Security	TBD
F0h - FFh	Vendor Specific	

8.1 CbCS SECURITY PROTOCOL

(New section in SPC-4: 6.29.3)

8.1.1 Overview

(New section in SPC-4: 6.29.3.1)

The SECURITY PROTOCOL IN command specifying the CbCS protocol requests the device server to return the security attributes of the:

a) logical unit; or

b) SCSI target device that contains the addressed SECURITY PROTOCOL well-known logical unit.

The command supports CbCS pages that may be requested one at a time. An application client requests a CbCS page by using a SECURITY PROTOCOL IN command with the SECURITY PROTOCOL field set to 07h (CbCS protocol) and the SECURITY PROTOCOL SPECIFIC field set to the requested CbCS page code.

The SECURITY PROTOCOL SPECIFIC field (see Table 6) specifies the CbCS pages.

Table 6 – SECURITY PROTOCOL SPECIFIC field

Code	Description	Support	Reference
0000h	SECURITY PROTOCOL IN supported CbCS page	M	8.1.2
0001h	SECURITY PROTOCOL OUT supported CbCS page	M	8.1.3
0002h-000Fh	Reserved		
0010h	Capabilities CbCS page	M	8.1.4
0011h	Attributes CbCS page	M	8.1.5
0012h	Set Master Key, Seed Exchange CbCS page	M	8.1.6
0013h – FFFFh	Reserved		
Support key: M – Mandatory for device servers that support the CbCS. O – Optional			

8.1.2 SECURITY PROTOCOL IN supported CbCS page

(New section in SPC-4: 6.29.3.2)

Table 7 specifies the format of the SECURITY PROTOCOL IN supported CbCS page.

Table 7 - SECURITY PROTOCOL IN supported CbCS page

Bit / Byte	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)	SECURITY PROTOCOL IN supported CbCS page (first)						
5								(LSB)
	. . .							
n-1	(MSB)	SECURITY PROTOCOL IN supported CbCS page (last)						
n								(LSB)

The SECURITY PROTOCOL IN supported CbCS page shall contain a list of all of the CbCS pages the device server supports for the SECURITY PROTOCOL IN command specifying the CbCS protocol in ascending order beginning with page code 0000h.

8.1.3 SECURITY PROTOCOL OUT supported CbCS page

(New section in SPC-4: 6.29.3.3)

Table 8 specifies the format of the SECURITY PROTOCOL OUT supported CbCS page.

Table 8 - SECURITY PROTOCOL IN supported CbCS page format

Bit / Byte	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)	SECURITY PROTOCOL IN supported CbCS page (first)						
5								(LSB)
	. . .							
n-1	(MSB)	SECURITY PROTOCOL IN supported CbCS page (last)						
n								(LSB)

The SECURITY PROTOCOL OUT supported CbCS page shall contain a list of all of the CbCS pages that the device server supports for the SECURITY PROTOCOL OUT command specifying the CbCS protocol in ascending order.

8.1.4 Capabilities CbCS page

(New section in SPC-4: 6.29.3.4)

Table 9 specifies the format of the Capabilities CbCS page.

Table 9 – Capabilities CbCS 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)
	. . .							
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)

Byte \ Bit	7	6	5	4	3	2	1	0
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 a single set of the CbCS master key and the CbCS working keys for the SCSI target device. A Global Keys Support (GKS) bit set to zero specifies that the device server does not support single set of the CbCS master key and the CbCS working keys for the SCSI target device.

A Logical Unit Keys Support (LUKS) bit set to one specifies that the device server supports separate sets of the CbCS master key and the CbCS working keys for each logical unit. A Logical Unit Keys Support (LUKS) bit set to zero specifies that the device server does not support separate sets of the CbCS master key and the CbCS working keys for each logical unit.

A Global Security Method Support (GSMS) bit set to one specifies that the SCSI target device that contains this logical unit supports global security method (i.e., contains a SECURITY PROTOCOL well known logical unit). A Global Security Method Support (GSMS) bit set to zero specifies that the device server requires the security methods to be assigned to each logical unit.

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

The SUPPORTED SECURITY METHOD fields contain coded values of the security methods (see 1.7) supported by the device server. The coded values are specified in Table 22.

The SUPPORTED INTEGRITY CHECK VALUE ALGORITHM fields contain coded values of the algorithm to compute integrity check values supported by the device server (see 1.8). The coded values are specified in [06-449r2 – Table K4 – Integrity algorithm identifiers].

The SUPPORTED DH GROUP attributes contain coded values identifying the supported values in the DH_GROUP field of Set Master Key, Seed Exchange page (see 9.1.4). The coded values are specified in [06-449r2 – Table K5 – Diffie-Hellman group identifiers].

8.1.5 Attributes CbCS page

(New section in SPC-4: 6.29.3.5)

Table 10 specifies the format of the Attributes CbCS page.

Table 10 - Attributes CbCS 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				
n								(LSB)

If the addressed logical unit is the SECURITY PROTOCOL well-known logical unit:

a) the SECURITY METHOD field is the security method assigned by the W-LUN's device manager to all logical units within the SCSI target device;

b) the POLICY ACCESS TAG field is the initial policy access tag assigned by the W-LUN's device server to all logical units within the SCSI target device;

c) if the device server does not support global security method (i.e., the GSMS bit is set to zero in the Capabilities CbCS page), then the SECURITY METHOD field is undefined; and

d) if the device server does not support global keys (i.e., the GKS bit is set to zero in the Capabilities CbCS page), then the MASTER KEY IDENTIFIER field and all the WORKING KEY IDENTIFIER fields should contain FFFF FFFFh

If the addressed logical unit is not the SECURITY PROTOCOL well-known logical unit:

a) the SECURITY METHOD field is the current security method used for the addressed logical unit;

b) the POLICY ACCESS TAG field is the current policy access tag assigned to the addressed logical unit;

c) if the device server does not support per-logical unit security method (i.e., the LUSMS bit is set to zero in the Capabilities CbCS page), then the SECURITY METHOD field is undefined; and

d) if the device server does not support per-logical unit keys (i.e., the LUKS bit is set to zero in the Capabilities CbCS page), the MASTER KEY IDENTIFIER field and all the WORKING KEY IDENTIFIER fields should contain FFFF FFFFh.

Security methods are described in detail in 1.7.

Secret keys are described in 2.2.

If secret keys are supported for the addressed logical unit, the values of those fields are as follows:

a) The MASTER KEY IDENTIFIER field shall contain the key identifier value from the most recent successful SECURITY PROTOCOL OUT command specifying the CbCS security protocol and the Set Master Key, Change Master Key page (see 9.1.5). If that command has never been processed, then the MASTER KEY IDENTIFIER field shall contain FFFF FFFEh; and

b) Each KEY IDENTIFIER field contains the key identifier value from the most recent successful SECURITY PROTOCOL OUT command specifying the CbCS security protocol and the Set Key page, with the KEY VERSION field set to the pertinent key (0-15) (see 9.1.3). If a secret key is invalid (e.g., never set, invalidated by a Set Master Key, Change Master Key page, or invalidated by a Set Key page), 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.

[Editor's 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 nexus or I_T_L nexus on which the SECURITY PROTOCOL IN command was sent. The security token shall be random as defined by RFC 1750. An I_T nexus loss event or reset event (see SAM-4) shall cause the security token to change.

Comment: Sivan: This is borrowed from OSD. Why do we need this to be random?

8.1.6 Set Master Key, Seed Exchange CbCS page

(New section in SPC-4: 6.29.3.6)

Table 11 specifies the format of the Set Master Key, Seed Exchange CbCS page.

Table 11 - Set Master Key, Seed Exchange CbCS page 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 Set Master Key, Seed Exchange CbCS page is received and no SECURITY PROTOCOL OUT command specifying Set Master Key, Seed Exchange CbCS page has been completed successfully 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 CbCS page on one I_T_L nexus may terminate the command with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SYSTEM RESOURCE FAILURE if any of the following command processing is incomplete on a different I_T_L nexus:

a) SECURITY PROTOCOL OUT command specifying Set Master Key, Seed Exchange CsCB page (see 9.1.4);

b) SECURITY PROTOCOL IN command specifying Set Master Key, Seed Exchange CsCB page (see 8.1.6); or

c) SECURITY PROTOCOL OUT command specifying Set Master Key, Change Master Key CsCB page (see 9.1.5).

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

a) A random number, y, is generated having a value between zero and DH_prime minus one observing the requirements in RFC 1750; and

b) The device server DH_data is equal to $DH\_generator^{y}$ modulo DH_prime.

The DH_generator and DH_prime values are identified by the Diffie-Hellman group specified in the DH GROUP field in the most recent SECURITY PROTOCL OUT command specifying the Set Master Key, Seed Exchange page (see 9.1.4) that was received on the same I_T_L nexus.

After GOOD status has been returned for SECURITY PROTOCOL OUT command and before the SECURITY PROTOCOL OUT command specifying the Set Master Key, Change Master Key page is processed, the next CbCS authentication master key and next CbCS generation master key shall be computed as described in 3, using a seed value that is the concatenation of the following:

1) DH_generator^{xy} modulo DH_prime; and

(GP – what is x?)

2) The whole content of the Device Identification VPD page (83h) returned from the addressed logical unit for the INQUIRY command (see 7.6.3 [spc4r09]).

9 Changes in SECURITY PROTOCOL OUT command

[Changes in section 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	CbCS	
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	

9.1 CbCS SECURITY PROTOCOL

(New section in SPC-4: 6.30.1)

9.1.1 Overview

The SECURITY PROTOCOL OUT command specifying CbCS protocol is used to configure the CbCS secret keys and attributes in the device server.

The command supports CbCS pages that may be sent one at a time. An application client requests to send a CbCS page by using a SECURITY PROTOCOL OUT command with the SECURITY PROTOCOL field set to 07h (CbCS protocol) and the SECURITY PROTOCOL SPECIFIC field set to the CbCS page code requested.

The SECURITY PROTOCOL SPECIFIC field (see Table 12) specifies the type of CbCS page that the application client is sending.

Table 12 - SECURITY PROTOCOL SPECIFIC field values

Code	Description	Support	Reference
0000h – 0010h	Reserved		
0011h	Set Attributes CbCS page	O	9.1.2
0012h	Set Key CbCS page	M	9.1.3
0013h	Set Master Key, Seed Exchange CbCS page	M	9.1.4
0014h	Set Master Key, Change Master Key CbCS page	M	9.1.5
0015h – FFFFh	Reserved		
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.

9.1.2 Set attributes CbCS page

(New section in SPC-4: 6.30.1.2)

Table 13 specifies the format of the Set Attributes CbCS page.

Table 13 – Set Attributes CbCS page format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (0011h)				
1								(LSB)
2	(MSB)			PAGE LENGTH (6)				
3								(LSB)
4	(MSB)			SECURITY METHOD				
5								(LSB)
6	(MSB)			POLICY ACCESS TAG				
9								(LSB)

The PAGE LENGTH field indicates the number of bytes of parameter data to follow. If the page length value is any value other than 6, 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 the security method to apply to the addressed logical unit (see 1.7). The SECURITY METHOD field is set to:

a) the reserved value FFFFh to specify no change shall be made to the current security method;

b) a value equal to the current security method shall not be considered an error; and

c) a value that does not match any of the supported security methods reported in the Capabilities CbCS page (see 8.1.4), shall cause the device server to terminate the SECURITY PROTOCOL OUT 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 list of coded values of security methods is defined in Table 22.

The POLICY ACCESS TAG field specifies a new policy access tag for the addressed logical unit. The value set to 0000 0000h specifies no change shall be made to the current policy access tag value. If the addressed logical unit is the SECURITY PROTOCOL well-known logical unit and the POLICY ACCESS TAG field contains any value other than 0000 0000h, the device server shall terminate the SECURITY PROTOCOL OUT command with CHECK CONDITION status and set the sense key to ILLEGAL REQUEST and the additional sense code to INVALID FIELD IN PARAMETER DATA.

This command shall be authorized and shall be sent encapsulated by a CbCS encapsulation (see 6).

9.1.3 Set Key CbCS page

(New section in SPC-4: 6.30.1.3)

Table 14 specifies the Set Key CbCS page format.

Table 14 - Set Key CbCS page format

Byte \ Bit	7	6	5	4	3	2	1	0
0	(MSB)			PAGE CODE (0012h)				
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 key version to be updated.

The KEY IDENTIFIER field specifies a unique identifier to be associated with the new secret key. The secret key identifier value shall be associated with the attribute specified in the Attributes CbCS page (see 8.1.5).

The SEED field contains a random number generated from a good source of entropy (e.g., as described in RFC 1750).

The updated secret key value shall be computed as described in 3.

This command shall be authorized and shall be sent encapsulated with CbCS encapsulation (see 6).

9.1.4 Set Master Key, Seed Exchange CbCS page

(New section in SPC-4: 6.30.1.4)

Table 15 specifies the Set Master Key, Seed Exchange CbCS page format.

Table 15 - Set Master Key CbCS 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 CbCS page on one I_T_L nexus may terminate the command with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SYSTEM RESOURCE FAILURE if any of the following command processing is incomplete on a different I_T_L nexus:

a) SECURITY PROTOCOL OUT command specifying Set Master Key, Seed Exchange CsCB page (see 9.1.4);

b) SECURITY PROTOCOL IN command specifying Set Master Key, Seed Exchange CsCB page (see 8.1.6); or

c) SECURITY PROTOCOL OUT command specifying Set Master Key, Change Master Key CsCB page (see 9.1.5).

The DH GROUP field contains the Diffie-Hellman group (see 06-449r2 - 7.7.4.11.1.4 Diffie-Hellman Group (D-H) identifiers) that identifies the DH_generator value and DH_prime value to be used in the seed exchange. If the value in the DH GROUP field is not listed in one of the SUPPORTED D-H GROUP fields in the Capabilities CbCS page (see 8.1.4), then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

The DH DATA LENGTH field specifies the number of bytes of the DH DATA field.

The DH_DATA field contains the DH data and is computed as follows:

DH data = $DH\_generator^{x}$ modulo DH_prime

Where:

X	is a value between zero and DH_prime minus one as defined in RFC 1750;
DH_generator	is defined by the DH GROUP field; and
DH_prime	is defined by the DH GROUP field.

9.1.5 Set Master Key, Change Master Key CbCS page

(New section in SPC-4: 6.30.1.5)

Table 16 specifies the format of the Set Master Key, Change Master Key CbCS page.

Table 16 - Set Master Key, Change Master Key CbCS 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								

If a SECURITY PROTOCOL OUT command specifying Set Master Key, Change Master Key CbCS page is received and no SECURITY PROTOCOL IN command specifying Set Master Key, Seed Exchange CbCS page has been completed successfully 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, Change master Key CbCS page on one I_T_L nexus may terminate the command with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to SYSTEM RESOURCE FAILURE if any of the following command processing is incomplete on a different I_T_L nexus:

a) SECURITY PROTOCOL OUT command specifying Set Master Key, Seed Exchange CsCB page (see 9.1.4);

b) SECURITY PROTOCOL IN command specifying Set Master Key, Seed Exchange CsCB page (see 8.1.6); or

c) SECURITY PROTOCOL OUT command specifying Set Master Key, Change Master Key CsCB page (see 9.1.5).

The KEY IDENTIFIER field specifies a unique identifier to be associated with the new CbCS master key. The secret key identifier value shall be placed into the MASTER KEY IDENTIFIER field in the Attributes CbCS page (see 8.1.5) before the command completes.

Table 17 specifies special secret key identifiers that shall not be used when setting secret keys. Any value not listed in the table is permitted for use by the application client when setting secret keys.

Table 17 – Special secret key identifiers

Value	Description
0000 0000h	A secret key that was never set or was invalidated
FFFF FFFEh	An initial secret key set by the device server
FFFF FFFFh	Pertinent secret key is not supported by the device server

If the value of the KEY IDENTIFIER field contains a value that is listed in Table 17, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

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 DH_data from the last SECURITY PROTOCOL OUT command specifying CbCS Set Master Key, Seed Exchange CbCS page on the same I_T_L nexus on which this command was received.

The DEVICE SERVER DATA LENGTH field contains the length in bytes of 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 CbCS page on the same I_T_L nexus on which this command was received.

If the content of the APPLICATION CLIENT DATA LENGTH field of the SECURITY PROTOCOL OUT commands Set Master Key, Seed Exchange CbCS page does not match the content of the:

a) DH DATA LENGTH field in a SECURITY PROTOCOL OUT Set Master Key, Seed Exchange CbCS page that was processed on this I_T_L nexus since a I_T nexus loss event, logical unit reset event, or reset event (see SAM-4);

b) DH DATA field in a SECURITY PROTOCOL OUT Set Master Key, Seed Exchange CbCS page that was processed on this I_T_L nexus since a I_T nexus loss event, logical unit reset event, or reset event;

c) PAGE LENGTH field in a SECURITY PROTOCOL IN Set Master Key, Seed Exchange CbCS page that was processed on this I_T_L nexus since a I_T nexus loss event, logical unit reset event, or reset event; or

d) DH DATA field in a SECURITY PROTOCOL IN Set Master Key, Seed Exchange CbCS page that was processed on this I_T_L nexus since a I_T nexus loss event, logical unit reset event, or reset event,

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

On successful completion of a SECURITY PROTOCOL OUT command specifying the CbCS protocol and the Set Master Key, Change Master Key CbCS page, the device server shall:

a) Replace the CbCS authentication master key with the next CbCS authentication master key computed after the return of GOOD status for the most recent SECURITY PROTOCOL IN command specifying the CbCS protocol and the Set Master Key, Seed Exchange CbCS page (see 8.1.6);

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

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

For every secret key that is invalidated by this command, the associated key identifier attribute shall have its attribute set to zero.

The next CbCS authentication master key computed after the return of GOOD status for the most recent SECURITY PROTOCOL IN command specifying the CbCS protocol and the Set Master Key, Seed Exchange CbCS page (see 8.1.6) shall be used to compute the capability key for this command.

10 RECEIVE CREDENTIAL command

[A new sub-section in section 6 of SPC-4]

The RECEIVE CREDENTIAL command (see Table 18) allows the application client to receive an CbCS credential for use in the CbCS encapsulation (see 6).

Table 18 - RECEIVE CREDENTIAL CDB format

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (7Fh)							
1	CONTROL							
2	Reserved							
6								
7	ADDITIONAL CDB LENGTH (n-7)							
8	(MSB)			SERVICE ACTION (1800h)				
9								(LSB)
10	(MSB)			ALLOCATION LENGTH				
11								(LSB)
12	Designation descriptor							
n								

If a RECEIVE CREDENTIAL command is received before a SECURITY PROTOCOL IN command has completed successfully on the same I_T_L nexus as the RECEIVE CREDENTIAL command was received with following field settings:

a) SECURITY PROTOCOL field set to xxh (i.e., IKEv2-SCSI); and

b) the SECURITY PROTOCOL SPECIFIC field set to 0103h (i.e., authentication phase),

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

[Editor's: note: The above paragraph is based on 06-449r2 and depends on it.]

The ALLOCATION LENGTH field is defined in 4.3.4.6 [spc4r09].

The format of the Designation descriptor field is defined in Table 338 [spc4r09]. The size of the Designation descriptor shall not exceed 24 bytes.

[Editor's note: The ESC proposal defines fixed size encapsulations, so the Capability size must fixed. Therefore, we have to put an upper bound on the designation descriptor length. 24 bytes allows for NAA-16 and for SCSI name string format using 16 byte length names concatenated to "naa."]

10.1 RECEIVE CREDENTIAL parameter data

The RECEIVE CREDENTIAL parameter data is defined in Table 19.

Table 19 - Credential format

Bit / Byte	7	6	5	4	3	2	1	0
0	CRED PRSNT	Reserved			CREDENTIAL FORMAT (1h)			
1	Reserved							
2	(MSB)	CREDENTIAL LENGTH (n-3)						
3								(LSB)
4	(MSB)	CAPABILITY LENGTH (k-5)						
5								(LSB)
6	CbCS capability descriptor							
k								
k+1	(MSB)	INTEGRITY CHECK VALUE LENGTH (n-(k+4))						
k+4								(LSB)
k+5	INTEGRITY CHECK VALUE							
n								

If the Credential Present (CRED PRSNT) bit is set to zero, no CbCS credential shall be returned, and the rest of the parameter data is undefined. If the Credential Present (CRED PRSNT) bit is set to one, an CbCS credential is returned in the parameter data.

The CREDENTIAL FORMAT field specifies the format of the CbCS credential. It shall be set to 1h.

[Editor's note: The intention of this field is to allow for future expansion of the standard to include multiple credential formats. The rest of the fields will depend on the value of this field. However, since it is currently the only format, we define the rest of the fields in the general credential form and not in a format-specific form.]

The CREDENTIAL LENGTH field indicates the length in bytes of the rest of the data (i.e. the CbCS credential) that follows.

The CAPABILITY LENGTH specifies the length in bytes of the CbCS capability descriptor field.

The format of the CbCS capability descriptor is defined in 10.1.1.

INTEGRITY CHECK VALUE LENGTH specifies the length of the INTEGRITY CHECK VALUE field.

The INTEGRITY CHECK VALUE field contains a value that the application client shall use for preparing CbCS credentials (see 1.8.2).

10.1.1 CbCS capability format

The format of the CbCS capability descriptor is defined in Table 20.

Table 20 - Capability descriptor format

Byte \ Bit	7	6	5	4	3	2	1	0
0	CAPABILITY FORMAT (1h)				KEY VERSION			
1	SECURITY METHOD							
2	(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	Designation descriptor							
63								
64	(MSB)	LBA START						
71								(LSB)
72	(MSB)	LENGTH						
79								(LSB)

The CAPABILITY FORMAT field (see Table 21) specifies the format of the CbCS capability. The CbCS capability format may also be the CbCS credential format. This standard only supports the CAPABILITY FORMAT field being set to 1h (i.e., the format defined by this standard).

Table 21 – Capability format field

Code	Description
0h	Reserved
1h	The format defined by this standard
2h - Fh	Reserved

If the CAPABILITY FORMAT field contains any value other than the ones defined in Table 21, then this 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 KEY VERSION field specifies which secret key, from the set of CbCS working keys, is being used to compute the integrity check value.

The SECURITY METHOD field should be set a valid CbCS security method (see Table 22). The CbCS security methods are described in detail in 1.7.

Table 22 – The CbCS Security methods

Code	Security Method	Description
0000h	NOSEC	No security (see 1.7.3)
0001h	CAPKEY	Integrity of CbCS capabilities (see 1.7.4)
0002h – 0FFFh	Reserved	
1000h – FFFEh	Vendor specific	
FFFFh	Reserved	

The INTEGRITY CHECK VALUE ALGORITHM field specifies the algorithm used to compute the integrity check value for this CbCS capability (see 1.8). It shall be set to a value defined in [06-449r2 – Table K4 – Integrity algorithm identifiers].

The CAPABILITY EXPIRATION TIME field specifies expiration time of the CbCS 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 set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

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

The AUDIT field contains a vendor specific value.

The PERMISSIONS BIT MASK descriptor (see Table 23) specifies the permissions allowed by this CbCS 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. Every bit corresponds to a command function (**Error! Reference source not found.**).

The association of command functions to SCSI commands is defined in 4.1.1 for commands defined in this standard. Associations for specific command sets are defined in the specific command set standards.

In Table 23 byte 0 and byte 1 specify the command functions for all SCSI commands. In Table 23 byte 2 and byte 3 may be used by a device type specific command set standard to specify command functions unique to the device type. Other command set standards shall not override the definition of Table 23 byte 0 and byte 1 as defined in this standard. The associations between the command functions specified in the permissions bit mask descriptor and SCSI commands defined in this standard are specified in 4.1.1.

Table 23 – PERMISSIONS BIT MASK descriptor

Byte \ Bit	7	6	5	4	3	2	1	0
0	DATA READ	DATA WRITE	ATTR READ	ATTR WRITE	SEC MGMT	RESRV	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.

An 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.

An 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 SEC MGMT bit set to one indicates the encapsulated SCSI command has security management permission.

A reservation (RESRV) bit set to zero indicates the encapsulated SCSI command has no persistent reservation permission. A SEC MGMT bit set to one indicates the encapsulated SCSI command has persistent reservation permission.

If the POLICY ACCESS TAG field contains a value other than zero, the policy access tag attribute of the logical unit (see 8.1.5) is compared to the POLICY ACCESS TAG field contents as part of verifying the capability. If the POLICY ACCESS TAG field contains zero, then no comparison is made to the policy access tag attribute of the logical unit. The CbCS management application client changes the policy access tag to prevent unsafe or temporarily undesirable accesses to a logical unit (see **Error! Reference source not found.**).

If the non-zero value in the CDB POLICY ACCESS TAG field is not identical to the value in the policy access tag attribute of the logical unit (see 8.1.5), then 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 format of the Designation descriptor field is defined in Table 338 [spc4r09]. The size of the Designation descriptor shall not exceed 24 bytes.

[Editor's note: The ESC proposal defines fixed size encapsulations, so the Capability size must fixed. Therefore, we have to put an upper bound on the designation descriptor

length. 24 bytes allows for NAA-16 and for SCSI name string format using 16 byte length names concatenated to “naa.”]

The LBA START field and the LENGTH field specify a range of logical block addresses to which the capability applies. If the value of the LENGTH field is zero, then the capability applies to the range of logical blocks starting at the address specified in LBA START and ending at the last block of the device. If logical block addresses are not applicable for the addressed logical unit type, then both the LBA START field and the LENGTH field shall be set to zero.

(GP – Security should only be logical unit based not LBA based. This option should be deleted)

11 PERFORM TASK MANAGEMENT FUNCTION command

[A new sub-section in section 6 of SPC-4]

(GP – This is a radical change in the way TMFs are sent to SCSI devices)

The PERFORM TASK MANAGEMENT FUNCTION command (see Table 24) allows a SAM-4 task management function (e.g., ABORT TASK) to be processed.

If this command is received and the CbCS bit in the extended inquiry data is set to zero for the logical unit, 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 COMMAND OPERATION CODE.

Table 24 - PERFORM TASK MANAGEMENT FUNCTION command format

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (XXh)							
1	TASK MANAGEMENT FUNCTION							
2	TASK TAG							
9								

The TASK MANAGEMENT FUNCTION field (see Table 25) specifies the SAM-4 task management function to be processed.

Table 25 - Task management function values

Value	SAM-4 Task Management Function	Task Tag Specified
01h	ABORT TASK	Yes
02h	ABORT TASK SET	No
04h	CLEAR TASK SET	No
08h	LOGICAL UNIT RESET	No
40h	CLEAR ACA	No

Value	SAM-4 Task Management Function	Task Tag Specified
80h	QUERY TASK	Yes

The format of the task tag is specified in the applicable SCSI transport protocol standard and the length of the task tag may be less than eight bytes. Any bytes between the end of the task tag and the end of the TASK TAG field shall be ignored (e.g., a two-byte task tag occupies the first two bytes of the TASK TAG field and the remaining six bytes are ignored).

12 Misc. changes

12.1 Change in C.3.5 Variable length CDB service action codes

The variable length CDB service action codes assigned by this standard are shown in table C.8.

Service Action Code	Description
1800h	RECEIVE CREDENTIAL
1801h – 1FFFh	Reserved

12.2 Change in Table 48 — Commands for all device types

In section 6.1, add the RECEIVE CREDENTIAL command to table 48.

13 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?

14 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, Noam Rinetzky, 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.