ENDL
TEXAS

Date: 27 June 2004
To: T10 Technical Committee and SNIA OSD TWG
From: Ralph O. Weber
Subject: Rewrite of OSD Security and Policy/Storage Manager Models

Revision History

This document is being presented in a 'build up' fashion. That is r0 contains an initial amount of changes and each revision after that contains additional details and changes.

r0 Shows only the model split between Capabilities (aka Policy/Storage Management) and Credentials (aka Security) and provides a rudimentary description for fencing
r1 Contains the same normative text as r0, but the strikeouts are removed
r2 Example configuration figure cleaned up and used blue lines for connections that do not use the service delivery subsystem. Moves the communications security requirements to the security manager Trust Assumptions subclause. Updates the security threats table to cover non-secure channels. Adds prototype CDB definition with changes resulting from the model split between Capabilities (aka Policy/Storage Management) and Credentials (aka Security).
r3 Makes all agreed changes that the author could locate on the SNIA OSD TWG reflector and a few from other sources. It is believe that this revision addresses all the 04-108 comments that identify a revision of this document as detailing the response to a T10 OSD Letter Ballot comment.

Summary

The document shows all the changes to OSD r09 needed to:

- Separate Capability handling (aka Policy/Storage Management) from Credential handling (aka Security)
- Treat security manager communications requirements as a trust assumption
- Modify threats analysis to cover CAPKEY over a non-secure channel

Changes made by this document are shown as red text additions and ~~red text removals~~.

The author believes that this revision of this document addresses the following previously unresolved OSD Letter Ballot comments:

- ENDL 2, HP 62, IBM 42, IBM 141, Seagate 8, and Veritas 65 motivated the development of this document, meaning the all the changes it proposes are addressed to those comments more or less
- Agilent 7 and two-thirds of EMC 8 should be resolved by moving the Partition_ID to the capability (as described in the response to IBM 141)
- EMC 1 should be addressed by the changes in 4.9.2 (Trust assumptions)
- EMC 2 should be addressed by the paragraph added at the end of 4.9.6.2
- EMC 3 should be addressed by the content of 4.9.m
- EMC 4 should be addressed by the changes in the "Security methods and threats thwarted" table in 4.9.3.1 (Security methods … Introduction)
- EMC 5, EMC 6, HP 33, HP 35, and Lingua 19 should be resolved the completely new definition of a security token (see 4.9.3.3)
- EMC 7, EMC 9, and part of Panasas 2 should be addressed the addition of step 5 to 4.9.5.2. The remainder of Panasas 2 should be addressed by the changes in 4.x.2.2.

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

- EMC 10 should be addressed by removing discussion of far-in-the-future nonces and making 'oldest valid nonce' and 'newest valid nonce' settable attributes (see 4.9.6.2 and 7.1.2.21)
- EMC 13 should be addressed by the changes to the SET MASTER KEY command and subclauses that it references (see 6.20)
- END 2 and HP 62 should be addressed by the new 4.x.3 as well as by the changes in 7.1.2.21, 7.1.2.22, and 7.1.2.23
- HP 51 and Intel 21 should be addressed by the global change of 'drive key' to 'root key'
- IBM 147, Panasas 5, and Seagate 9 should be resolved by the addition of a SECURITY METHOD field to the Capability (see 4.x.2.2 and 4.9.m)
- Seagate 18 should be resolved by the changes in 4.9.7, table 16, table 17, table 23, table 43, and table 44.

This document also shows all approved changes for the clauses modified by this document, specifically: Agilent 3, Agilent 8, Agilent 9, Agilent 10, Brocade 17, EMC 12, HP 30, HP 31, HP 32, HP 34, HP 36, HP 40, HP 42, HP 43, HP 47, HP 48, HP 49, HP 50, HP 54, HP 55, HP 56, HP 57, HP 58, HP 59, HP 76, HP 77, HP 85, HB 115, HP 120, HP 122, IBM 39, IBM 40, IBM 41, IBM 43, IBM 44, IBM 46, IBM 47, IBM 48, IBM 49, IBM 50, IBM 52, IBM 53, IBM 55, IBM 56, IBM 57, IBM 58, IBM 59, IBM 91, IBM 128, IBM 129, IBM 130, IBM 131, IBM 132, IBM 133, IBM 134, IBM 136, IBM 137, IBM 138, IBM 142, IBM 153, Lingua 20, Lingua 22, Lingua 23, Lingua 24, Lingua 25, Lingua 26, Lingua 27, Lingua 28, Lingua 30, Lingua 31, Lingua 32, Lingua 36, Lingua 37, LSI 12, Seagate 7, Seagate 10, Seagate 11, Seagate 13, Seagate 14, Seagate 15, Seagate 18, Seagate 19, Seagate 20, Seagate 21, Seagate 50, Seagate 51, Veritas 45, Veritas 47, Veritas 49, Veritas 50, Veritas 51, Veritas 52, Veritas 53, Veritas 57, Veritas 58, Veritas 59, Veritas 61, Veritas 62, Veritas 63, Veritas 64, Veritas 66, Veritas 67, Veritas 68, Veritas 69, Veritas 70, Veritas 71, Veritas 72, Veritas 107, Veritas 108, Veritas 117, Veritas 118, Veritas 119, and Veritas 120.

These changes are shows as blue text additions and ~~blue text removals~~.

Suggested Changes:

2.4 Approved IETF References

Copies of the following approved IETF standards may be obtained through the Internet Engineering Task Force (IETF) at www.ietf.org.

RFC 1750, Randomness Recommendations for Security

RFC 2401, Security Architecture for the Internet Protocol

RFC 2409, The Internet Key Exchange

RFC 3526, More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange

3.2 Acronyms

…

DH Diffie-Hellman (see 2.4, RFC 2409 and RFC 3526)

…

IANA Internet Assigned Numbers Authority (see www.iana.org)

…

MODP Modular Exponential (see 2.4, RFC 3526)

…

4.4 Elements of the example configuration

The example in this subclause (see figure 1) illustrates the three mandatory and two optional constituents of an OSD configuration:

a) Object-Based Storage Devices;
b) Service delivery subsystem;
c) Host systems (i.e., initiator devices);
d) Optionally, a security manager; and
e) Optionally, a policy/storage manager.

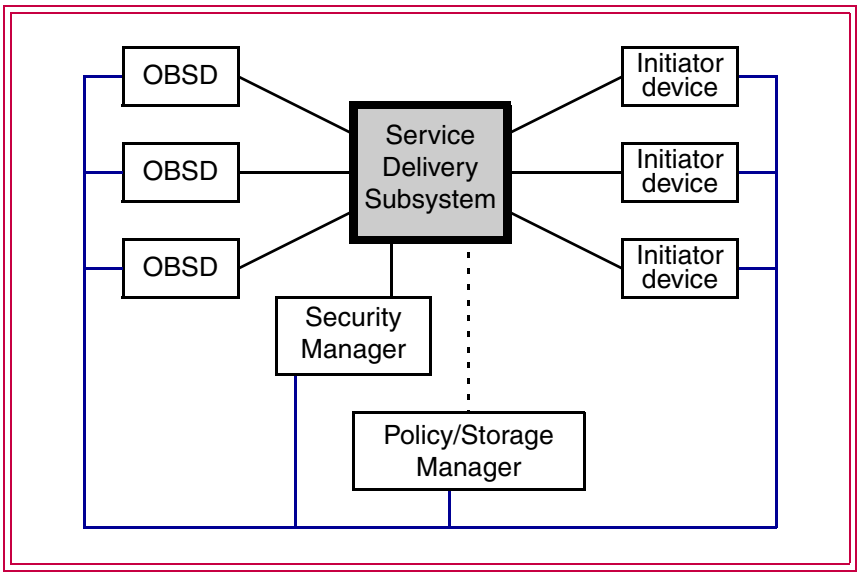

Figure 1 — Example OSD Configuration

The OBSDs are the storage components of the system to be shared (e.g., disc drives, RAID subsystems, tape drives, tape libraries, optical drives, jukeboxes, or other storage devices).

Application clients using multiple SCSI initiator ports share ~~and~~ directly access an OBSD (see 3.1.26) via the service delivery subsystem. The service delivery subsystem is used by the components in the OSD model, except possibly the policy/storage manager and/or security manager, to intercommunicate. The OSD security model (see 4.9) does not require the service delivery subsystem to provide security-related services (i.e., authentication and confidentiality ~~data privacy~~), but is designed to take advantage of whatever security-related services are provided.

The policy/storage manager (see 4.x), if present, coordinates access constraints between OSD device servers and application clients, preparing the capabilities application clients place in CDBs to gain access to OSD objects and command functions. ~~maintains storage policies in a manner that is outside the scope of this standard.~~

The security manager (see 4.9), if present, ~~coordinates access~~ secures capabilities in cryptographically protected credentials ~~between~~ for OSD device servers and application clients.

The policy/storage manager and security manager may reside in the OBSDs, in applications clients, or as a separate entities.

The policy/storage manager and security manager may use the service delivery subsystem and be an application client, but they ~~the security manager~~ also may use another mechanism to communicate with the OSD device servers and application clients. Security-related requirements on the communications mechanisms used by the security manager are described in 4.9.2

4.x Policy/storage management

4.x.1 Overview

The policy/storage manager:

a) Provides access policy controls to application clients via preparation of policy-coordinated capabilities (see 4.x.2); and
b) In concert with the OSD logical unit, prevents unsafe or temporarily undesirable utilization of OBSD storage (see 4.x.3).

4.x.2 Capabilities

4.x.2.1 Introduction

Each CDB defined by this standard includes a capability (see 4.x.2.2) whose contents specify the command functions (see 3.1.10) that the device server is allowed to process in response to the command.

The device server validates that the requested command functions are allowed ~~permitted~~ by the capability based on:

a) The type of functions (e.g., read, write, attributes setting, attributes retrieval); and
b) The OSD object on which the command functions are to be ~~performed~~ processed.

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

The policy/storage manager shall coordinate the delivery of capabilities to application clients with the security manager (see 4.9) as follows:

a) If the security method for all partitions in the OSD logical unit is NOSEC (see 4.9.3.2), then the policy/storage manager may:
   A) Allow application clients to prepare their own capabilities;
   B) Coordinate the preparation of capabilities for multiple application clients in response to requests, the format and transport mechanisms for which are outside the scope of this standard; or
   C) Coordinate the preparation of capabilities with the security manager as described in item b);
   or
b) If a security method other than NOSEC is in use by any partition in the OSD logical unit, then the policy/storage manager shall coordinate the preparation of capabilities with the security manager by:
   A) Requiring application clients to request credentials and capabilities from the security manager; and
   B) Preparing capabilities only is response to requests from the security manager.

4.x.2.2 Capability format

4.x.2.2.1 Introduction

A capability (see table 1) ~~is the portion of a credential (see 4.9.4.1) that~~ is included in a CDB to enable the device server to verify that the sender is allowed to perform the ~~function~~ command functions (see 3.1.10) specified ~~described~~ by the CDB ~~command~~.

Table 1 — Capability format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved				~~CREDENTIAL/~~CAPABILITY FORMAT (1h)			
1	KEY VERSION				INTEGRITY CHECK VALUE ALGORITHM			
2	SECURITY METHOD							
3	Reserved							
4	(MSB)			CAPABILITY EXPIRATION TIME				
9								(LSB)
10				AUDIT				
29								
30	(MSB)			CAPABILITY DISCRIMINATOR				
41								(LSB)
42	(MSB)			OBJECT ~~CREATION~~ CREATED TIME				
47								(LSB)
48	OBJECT TYPE							
49				PERMISSIONS BIT MASK				
53								
54	Reserved							
55	OBJECT DESCRIPTOR TYPE				Reserved			
56				OBJECT DESCRIPTOR				
79								

The ~~CREDENTIAL/~~CAPABILITY FORMAT field (see table 2) specifies the format of the ~~credential and~~ capability. If capabilities are coordinated with the security manager, the capability format also is the credential format. The policy/storage manager shall set the CAPABILITY FORMAT field to 1h (i.e., the format defined by this standard).

Table 2 — C~~redential/c~~apability format values

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

If the CAPABILITY FORMAT field contains 1h, the device server shall verify that the command functions requested by a CDB are permitted by the capability as described in this subclause. The device server may verify that a command

function is permitted after other command functions are completed. The device server shall verify that a command function is permitted before any part of the command function is performed. (E.g., the device server may delay verifying that the set attributes command functions specified by a set attributes list are allowed until the requested read command function is completed, but all the capability permissions concerning the setting attributes are to be verified before any attribute values are changed.)

The KEY VERSION field, INTEGRITY CHECK VALUE ALGORITHM field, and SECURITY METHOD field are used by the security manager. If capabilities are not coordinated with the security manager, the KEY VERSION field, INTEGRITY CHECK VALUE ALGORITHM field, and SECURITY METHOD field are reserved.

If CDB contains a non-zero value in the SECURITY METHOD field, the integrity of the CDB shall be validated (see 4.9.5.2) before any other command processing actions are undertaken (i.e., before verifying that command functions requested in the CDB are permitted by the capability).

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 CDB SECURITY METHOD field or CAPABILITY FORMAT field contains zero and one of the following is true:

a) The command is SET KEY (see 6.23) or SET MASTER KEY (see 6.24); or
b) The security method attribute in the Partition Policy/Security attributes page (see 7.1.2.21) specifies a security method other than NOSEC for the partition identified as follows:
   A) For the CREATE PARTITION command (see 6.7), FLUSH OSD command (see 6.y), FORMAT OSD command (see 6.9), the identified partition is partition zero (see 3.1.31);
   B) For any command that is not one of those already listed, the partition is identified by the contents of the CDB PARTITION_ID field.

The CAPABILITY EXPIRATION TIME field specifies the value of the clock attribute in the Root Information attributes page (see 7.1.2.8) after which this capability is no longer valid. If a CDB CAPABILITY EXPIRATION TIME field contains a value other than zero and the value of the clock attribute in the Root Information attributes page is greater than the value in the CAPABILITY EXPIRATION TIME field, 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.

Successful use of the capability expiration time requires some degree of synchronization between the clocks of the device server, policy/storage manager, and security manager. The protocol for synchronizing the clocks is outside the scope of this standard.

The AUDIT field is a vendor specific value that the policy/storage manager and/or security manager may use to associate the capability ~~and credential~~ with a specific application client.

The CAPABILITY DISCRIMINATOR field contains a nonce (see 3.1.23) that differentiates one capability ~~and credential~~ from another.

~~The OBJECT CREATION TIME field specifies the contents of the creation time attribute in the User Object Timestamps attributes page (see 7.1.2.18), Collection Timestamps attributes page (see 7.1.2.17), Partition Timestamps attributes page (see 7.1.2.16), or Root Timestamps attributes page (see 7.1.2.15), for the OSD object to which the credential applies. A value of zero specifies that any object creation time is allowed.~~

The OBJECT ~~CREATION~~ CREATED TIME field specifies the contents of the ~~creation~~ created time attribute for the OSD object (see table 3) to which the ~~credential~~ capability applies. A value of zero specifies that any object ~~creation~~ created time is allowed.

Table 3 — ~~Creation~~ Created time for OSD objects by type

Object Type (see table 4)	**Attributes page containing ~~creation~~ created time attribute to which ~~credential~~ the capability OBJECT ~~CREATION~~ CREATED TIME field is applies**
ROOT	~~Root Timestamps attributes page (see 7.1.2.15)~~ Partition Timestamps attributes page (see 7.1.2.16) for partition zero (see 3.1.31)
PARTITION	Partition Timestamps attributes page (see 7.1.2.16)
COLLECTION	Collection Timestamps attributes page (see 7.1.2.17)
USER	User Object Timestamps attributes page (see 7.1.2.18)

If a CDB OBJECT CREATED TIME field contains a value other than zero and the value in the OBJECT ~~CREATION~~ CREATED TIME field is not identical to the value in the ~~creation~~ created time attribute from the associated timestamps attributes page (see table 3), 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 OBJECT TYPE field (see table 4) specifies the type of OSD object to which this capability allows access ~~and aids in the determination of how to validate the credential and capability~~. If capabilities are coordinated with the security manager, the OBJECT TYPE field is used to select the secret key that is used in validating the credential.

Table 4 — Object type values

Value	**Name**	**~~OSD~~ OSD object type to which access is allowed**
01h	ROOT	Root object
02h	PARTITION	Partition
40h	COLLECTION	Collection
80h	USER	User objects
all other values	Reserved	

If the command functions specified by the CDB are not allowed for the OSD object type specified in the CDB OBJECT TYPE field (see 4.x.2.2), 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 PERMISSIONS BIT MASK field (see table 5) specifies which functions are allowed by this capability. More than one permissions bit may be set within the constraints specified in 4.x.2.3 resulting in a single ~~credential and~~ capability that allows more than one command function.

Table 5 — Permissions bit mask format

Bit / Byte	7	6	5	4	3	2	1	0
49	READ	WRITE	GET_ATTR	SET_ATTR	CREATE	REMOVE	OBJ_MGMT	~~Reserved~~ APPEND
50	DEV_MGMT	GLOBAL	~~SECURITY~~ POL/SEC	Reserved				
51	Reserved							
52	Reserved							
53	Reserved							

A READ bit set to one allows read access to the data in an OSD object, but not to the attributes. For the root object, partitions, and collections the data in the OSD object is the list of other objects contained in the OSD object. A READ bit set to zero prohibits read access to the data in an OSD object.

A WRITE bit set to one allows processing of the WRITE command (see 6.21) ~~write access to the data in a user object~~, but not ~~to the~~ access to user object attributes. A WRITE bit set to zero prohibits processing of the WRITE command ~~write access to the data in a user object~~.

A GET_ATTR (get attributes) bit set to one allows retrieval of (i.e., read access to) the attributes associated with an OSD object. A GET_ATTR bit set to zero prohibits retrieval of attributes except for the attributes in the Current Command attributes page (see 7.1.2.24).

A SET_ATTR (set attributes) bit set to one allows the setting of (i.e., write access to) the attributes associated with an OSD object except for attributes located in the OSD object's policy/security attributes page (e.g., the User Object Policy/Security attributes page (see 7.1.2.23) if the OSD object is a user object). The setting of attributes located in the OSD object's policy/security attributes page is allowed only if both the SET_ATTR bit and the ~~SECURITY~~ POL/SEC bit are set to one. A SET_ATTR bit set to zero prohibits the setting of the attributes associated with an OSD object.

A CREATE bit set to one allows the creation of OSD objects. A CREATE bit set to zero prohibits the creation of OSD objects.

A REMOVE bit set to one allows the removal of OSD objects. A REMOVE bit set to zero prohibits the removal of OSD objects.

An OBJ_MGMT (object management) bit set to one allows command functions that may change how the OSD logical unit handles an OSD object without affecting the stored data, stored attributes, commands in the task set, policies, or security for the OSD object. A OBJ_MGMT bit set to zero prohibits such command functions.

An APPEND bit set to one allows processing of the APPEND command (see 6.2), but not access to user object attributes. A APPEND bit set to zero prohibits processing of the APPEND command.

A DEV_MGMT (device management) bit set to one allows command functions that affect the OSD logical unit. A DEV_MGMT bit set to zero prohibits command functions that affect the OSD logical unit.

A GLOBAL bit set to one allows command functions that may affect all the OSD objects in the OSD logical unit. A GLOBAL bit set to zero prohibits command functions that may affect all the OSD objects in the OSD logical unit.

A ~~SECURITY~~ POL/SEC bit set to one allows command functions that affect the policy/security functions performed for one or more OSD objects. A ~~SECURITY~~ POL/SEC bit set to zero prohibits command functions that affect the policy/security functions performed for one or more OSD objects.

If the command functions specified by the CDB are not allowed by the CDB PERMISSIONS BIT MASK field (see 4.x.2.2), 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 OBJECT DESCRIPTOR TYPE field (see table 6) specifies the format of information that appears in the OBJECT DESCRIPTOR field.

Table 6 — Object descriptor types

Object Descriptor Type	Name	Description	Reference
0h	NONE	The OBJECT DESCRIPTOR field shall be ignored	
1h	~~1~~OBJECT U/C	A single ~~partition,~~ collection~~,~~ or user object	4.x.2.2.2
2h	PAR	A single partition, including partition zero	4.x.2.2.3
~~2~~3h - Fh		Reserved	

4.x.2.2.2 U/C ~~1OBJECT~~ capability object descriptor

If the object descriptor type is ~~1OBJECT~~ U/C (i.e., 1h), the OBJECT DESCRIPTOR field shall have the format shown in table 7, specifying a single ~~partition,~~ collection~~,~~ or user object to which the capability allows access.

Table 7 — User object/collection object descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
56	(MSB)							
59				~~SECURITY VERSION~~ POLICY ACCESS TAG				(LSB)
60	(MSB)							
67				ALLOWED PARTITION_ID				(LSB)
68	(MSB)							
75				~~SINGLE~~ ALLOWED OBJECT_ID				(LSB)
76								
79				Reserved				

If the ~~SECURITY VERSION~~ POLICY ACCESS TAG field contains a value other than zero, the ~~security version~~ policy access tag attribute identified by the command and object type field (see table 8) is compared to the ~~SECURITY VERSION~~ POLICY ACCESS TAG field contents as part of verifying the capability. ~~the algorithm for validating credentials and capabilities described in 4.9.5.2.~~ If the ~~SECURITY VERSION~~ POLICY ACCESS TAG field contains zero, then no comparison is made to any ~~security version~~ policy access tag attribute. The policy/storage manager or OSD logical unit changes the policy access tag to prevent unsafe or temporarily undesirable accesses to an OSD object (see 4.x.3). ~~Changing the security version tag attribute is one way the security manager may invalidate credentials (see 4.9.5.5).~~

Table 8 — ~~Security version~~ Policy access tag OSD objects

Command	Object Type (see table 4)	Attributes page containing ~~security version~~ policy access tag attribute to which credential ~~SECURITY VERSION~~ POLICY ACCESS TAG field is compared
CREATE PARTITION	PARTITION	Partition Policy/Security attributes page (see 7.1.2.21) for partition zero (see 3.1.31)
CREATE COLLECTION	COLLECTION	Partition Policy/Security attributes page (see 7.1.2.21)
CREATE or CREATE AND WRITE	USER	Partition Policy/Security attributes page (see 7.1.2.21)
All other commands	ROOT	Partition Policy/Security attributes page (see 7.1.2.21) for partition ~~0h~~ zero
	PARTITION	Partition Policy/Security attributes page (see 7.1.2.21)
	COLLECTION	Collection Policy/Security attributes page (see 7.1.2.22)
	USER	User Object Policy/Security attributes page (see 7.1.2.23)

If the non-zero value in the CDB ~~SECURITY VERSION~~ POLICY ACCESS TAG field is not identical to the value in the ~~security version~~ policy access tag attribute from the associated security attributes page (see table 8), 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 ALLOWED PARTITION_ID field specifies the partition to which access is allowed. 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:

a) The ALLOWED PARTITION_ID field contains zero in a CDB; or
b) The ALLOWED PARTITION_ID field contents do not match the contents of the PARTITION_ID field in the CDB.

The ~~SINGLE~~ ALLOWED OBJECT_ID field specifies the ~~contains the Partition_ID (see 4.6.4),~~ Collection_Object_ID (see 4.6.6)~~,~~ or User_Object_ID (see 4.6.5) of the OSD object to which the capability allows access~~, with the type of OSD object being identified by the OBJECT TYPE field (see table 4)~~. 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:

a) The command is not CREATE, CREATE AND WRITE, or CREATE COLLECTION and the ALLOWED OBJECT_ID field contains zero;
b) The OBJECT TYPE field contains 40h (i.e., COLLECTION) and the ALLOWED OBJECT_ID field contents do not match the contents of the CDB COLLECTION_OBJECT_ID field or REQUESTED COLLECTION_OBJECT_ID field; or
c) The OBJECT TYPE field contains 80h (i.e., USER) and the ALLOWED OBJECT_ID field contents do not match the contents of the CDB USER_OBJECT_ID field or REQUESTED USER_OBJECT_ID field.

4.x.2.2.3 PAR capability object descriptor

If the object descriptor type is PAR (i.e., 2h), the OBJECT DESCRIPTOR field shall have the format shown in table 9, specifying a single partition, including partition zero (see 3.1.31), to which the capability allows access.

Table 9 — Partition descriptor format

Bit / Byte	7	6	5	4	3	2	1	0
56	(MSB)							
59				POLICY ACCESS TAG				(LSB)
60	(MSB)							
67				ALLOWED PARTITION_ID				(LSB)
68								
79				Reserved				

The POLICY ACCESS TAG field is described in 4.x.2.2.2.

The ALLOWED PARTITION_ID field specifies the partition to which access is allowed. 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:

a) The CDB USER_OBJECT_ID field, REQUESTED USER_OBJECT_ID field, COLLECTION_OBJECT_ID field or REQUESTED COLLECTION_OBJECT_ID field, if any, contains a value other than zero;
b) The OBJECT TYPE field contains 02h (i.e., PARTITION) and one of the following is true:
   A) The command is not CREATE PARTITION and the ALLOWED PARTITION_ID field contains zero;
   B) The ALLOWED PARTITION_ID field contents do not match the contents of the CDB PARTITION_ID field;
   or
c) The OBJECT TYPE field contains 01h (i.e., ROOT) and one of the following is true:
   A) The ALLOWED PARTITION_ID field contains a value other than zero; or
   B) The CDB PARTITION_ID field, if any, contains a value other than zero.

4.x.2.3 Capabilities ~~Credentials~~ and commands allowed

The validity of a specific command and some of the ~~function-related~~ command function (see 3.1.10) related fields in that command is determined by the presence of specific combinations of values in capability fields as shown in table 10. A command function is allowed if at least one row in table 10 allows it, even if a different row that applies does not allow it.

Any command may retrieve or set attributes. ~~and combinations~~ The combinations of capability fields that allow those functions are shown in table 11. Retrieving or setting attributes is allowed if at least one row in table 11 allows it, even if a different row that applies does not allow it.

A single ~~credential~~ capability for a single object type may allow processing of multiple command functions (e.g., read and write) as well as ~~the retrieval~~ retrieving and setting ~~of~~ attributes by combining the permission bits values described in multiple rows of table 10 and table 11.

Table 10 — Commands allowed by specific capability field values (Sheet 1 of 5)

Commands allowed and CDB fields whose contents are restricted by ~~credential~~ capability field contents, if any	Capability Field values that allow a command		
	Object Type Name	Permission Bits That Are Set To One	Object Descriptor Name
An APPEND command ~~with the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field and the CDB USER_OBJECT_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field.~~	USER	APPEND ~~WRITE~~	~~1OBJECT~~ U/C
A CREATE command ~~with the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field and the CDB REQUESTED USER_OBJECT_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field.~~	USER	CREATE	~~1OBJECT~~ U/C
~~A CREATE command with the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field, the CDB REQUESTED USER_OBJECT_ID field equal to zero.~~	~~USER~~	~~CREATE~~	~~NONE~~
A CREATE AND WRITE command ~~with the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field and the CDB REQUESTED USER_OBJECT_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field.~~	USER	CREATE and WRITE	~~1OBJECT~~ U/C
~~A CREATE AND WRITE command with the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field, the CDB REQUESTED USER_OBJECT_ID field equal to zero.~~	~~USER~~	~~CREATE and WRITE~~	~~NONE~~
Combinations of OBJECT TYPE field, PERMISSION BITS field, and OBJECT DESCRIPTOR TYPE field values not shown in this table and table 11 are reserved. The ~~credential and~~ capability fields not shown in this table may place additional limits on the objects that are allowed to be accessed.			
[a] ~~The object descriptor SINGLE OBJECT_ID field shall contain zero.~~			

Table 10 — Commands allowed by specific capability field values (Sheet 2 of 5)

Commands allowed and CDB fields whose contents are restricted by ~~credential~~ capability field contents, ~~if any~~	Capability Field values that allow a command		
	Object Type Name	Permission Bits That Are Set To One	Object Descriptor Name
A CREATE COLLECTION command ~~with the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field and the CDB REQUESTED COLLECTION_OBJECT_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field.~~	COLLECTION	CREATE	~~1OBJECT~~ U/C
~~A CREATE COLLECTION command with the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field and the CDB REQUESTED COLLECTION_OBJECT_ID field equal to zero.~~	~~COLLECTION~~	~~CREATE~~	~~NONE~~
A CREATE PARTITION command ~~with the CDB REQUESTED PARTITION_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field.~~	PARTITION	CREATE	~~1OBJECT~~ PAR
~~A CREATE PARTITION command with the CDB REQUESTED PARTITION_ID field equal to zero.~~	~~PARTITION~~	~~CREATE~~	~~NONE~~
A FLUSH ~~OBJECT~~ command ~~with the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field, the CDB USER_OBJECT_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field, and the USER_OBJECT_ID field not containing a Collection_Object_ID.~~	USER	OBJ_MGMT	~~1OBJECT~~ U/C
A FLUSH COLLECTION ~~OBJECT~~ command ~~with the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field, the CDB COLLECTION_OBJECT_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field, and the USER_OBJECT_ID field containing a Collection_Object_ID.~~	COLLECTION	OBJ_MGMT	~~1OBJECT~~ U/C
A FLUSH PARTITION ~~OBJECT~~ command ~~with the CDB PARTITION_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field and the CDB USER_OBJECT_ID field equal to zero.~~	PARTITION	OBJ_MGMT	~~1OBJECT~~ PAR
A FLUSH OSD ~~OBJECT~~ command ~~with the CDB PARTITION_ID field and the CDB USER_OBJECT_ID field both equal to zero or a FORMAT OSD command.~~	ROOT	OBJ_MGMT	~~1OBJECT~~ [a] PAR
Combinations of OBJECT TYPE field, PERMISSION BITS field, and OBJECT DESCRIPTOR TYPE field values not shown in this table and table 11 are reserved. The ~~credential and~~ capability fields not shown in this table may place additional limits on the objects that are allowed to be accessed.			
[a] ~~The object descriptor SINGLE OBJECT_ID field shall contain zero.~~			

Table 10 — Commands allowed by specific capability field values (Sheet 3 of 5)

Commands allowed and CDB fields whose contents are restricted by ~~credential~~ capability field contents, ~~if any~~	Capability Field values that allow a command		
	Object Type Name	Permission Bits That Are Set To One	Object Descriptor Name
A FORMAT OSD command	ROOT	OBJ_MGMT and GLOBAL	PAR
A GET ATTRIBUTES command addressed to a user object	USER	see table 11	U/C
A GET ATTRIBUTES command addressed to a collection	COLLECTION	see table 11	U/C
A GET ATTRIBUTES command addressed to a partition	PARTITION	see table 11	PAR
A GET ATTRIBUTES command addressed to the root object	ROOT	see table 11	PAR
A LIST command addressed to a partition ~~with the CDB PARTITION_ID field containing a value other than zero that matches the contents of the object descriptor SINGLE OBJECT_ID field.~~	PARTITION	READ	~~1OBJECT~~ PAR
A LIST command addressed to the root object ~~with the CDB PARTITION_ID field equal to zero.~~	ROOT	READ	~~1OBJECT~~ [a] PAR
A LIST COLLECTION command addressed to a collection ~~with the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field and the CDB COLLECTION_OBJECT_ID field containing a value other than zero that matches the contents of the object descriptor SINGLE OBJECT_ID field. If the object descriptor SINGLE OBJECT_ID field contains zero, a LIST COLLECTION command is allowed to list the collections defined in the partition.~~	COLLECTION	READ	~~1OBJECT~~ U/C
A LIST COLLECTION command addressed to a partition	PARTITION	READ	PAR
A PERFORM TASK MANAGEMENT command with function code of ABORT TASK or QUERY TASK addressed to a user object ~~, the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field, the CDB USER_OBJECT_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field, and the USER_OBJECT_ID field not containing a Collection_Object_ID.~~	USER	DEV_MGMT	~~1OBJECT~~ U/C

Combinations of OBJECT TYPE field, PERMISSION BITS field, and OBJECT DESCRIPTOR TYPE field values not shown in this table and table 11 are reserved.
The ~~credential and~~ capability fields not shown in this table may place additional limits on the objects that are allowed to be accessed.

[a] ~~The object descriptor SINGLE OBJECT_ID field shall contain zero.~~

Table 10 — Commands allowed by specific capability field values (Sheet 4 of 5)

Commands allowed and CDB fields whose contents are restricted by ~~credential~~ capability field contents, ~~if any~~	Capability Field values that allow a command		
	Object Type Name	Permission Bits That Are Set To One	Object Descriptor Name
A PERFORM TASK MANAGEMENT command with function code of ABORT TASK or QUERY TASK addressed to a collection ~~, the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field, the CDB USER_OBJECT_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field, and the USER_OBJECT_ID field containing a Collection_Object_ID.~~	COLLECTION	DEV_MGMT	~~1OBJECT~~ U/C
A PERFORM TASK MANAGEMENT command with function code of ABORT TASK or QUERY TASK addressed to a partition ~~, the CDB PARTITION_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field, and a CDB USER_OBJECT_ID field equal to zero.~~	PARTITION	DEV_MGMT	~~1OBJECT~~ PAR
A PERFORM TASK MANAGEMENT command with function code of ABORT TASK or QUERY TASK addressed to the root object ~~with function code of ABORT TASK or QUERY TASK and the CDB PARTITION_ID field and CDB USER_OBJECT_ID field both equal to zero.~~	ROOT	DEV_MGMT	~~1OBJECT~~ [a] PAR
A~~ny~~ PERFORM TASK MANAGEMENT command or a PERFORM SCSI COMMAND command~~.~~	ROOT	DEV_MGMT and GLOBAL	~~1OBJECT~~ [a] PAR
A READ command ~~with the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field and the CDB USER_OBJECT_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field.~~	USER	READ	~~1OBJECT~~ U/C
A REMOVE command ~~with the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field and the CDB USER_OBJECT_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field.~~	USER	REMOVE	~~1OBJECT~~ U/C
A REMOVE COLLECTION command ~~with the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field and the CDB COLLECTION_OBJECT_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field.~~	COLLECTION	REMOVE	~~1OBJECT~~ U/C
Combinations of OBJECT TYPE field, PERMISSION BITS field, and OBJECT DESCRIPTOR TYPE field values not shown in this table and table 11 are reserved. The ~~credential and~~ capability fields not shown in this table may place additional limits on the objects that are allowed to be accessed.			
[a] ~~The object descriptor SINGLE OBJECT_ID field shall contain zero.~~			

Table 10 — Commands allowed by specific capability field values (Sheet 5 of 5)

Commands allowed and CDB fields whose contents are restricted by ~~credential~~ capability field contents, ~~if any~~	Capability Field values that allow a command		
	Object Type Name	Permission Bits That Are Set To One	Object Descriptor Name
A REMOVE PARTITION ~~command with the CDB PARTITION_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field.~~	PARTITION	REMOVE	~~1OBJECT~~ PAR
A SET ATTRIBUTES command addressed to a user object	USER	see table 11	U/C
A SET ATTRIBUTES command addressed to a collection	COLLECTION	see table 11	U/C
A SET ATTRIBUTES command addressed to a partition	PARTITION	see table 11	PAR
A SET ATTRIBUTES command addressed to the root object	ROOT	see table 11	PAR
A SET KEY command with KEY TO SET field equal to 10b or 11b ~~and the CDB PARTITION_ID field containing a non-zero value that matches the contents of the object descriptor SINGLE OBJECT_ID field.~~	PARTITION	DEV_MGMT and ~~SECURITY~~ POL/SEC	~~1OBJECT~~ PAR
A~~ny~~ SET KEY command with KEY TO SET field equal to 01b ~~with the CDB PARTITION_ID field equal to zero.~~	ROOT	DEV_MGMT and ~~SECURITY~~ POL/SEC	~~1OBJECT~~ [~~a~~] PAR
A~~ny~~ SET MASTER KEY command~~.~~	ROOT	DEV_MGMT and ~~SECURITY~~ POL/SEC	~~1OBJECT~~ [~~a~~] PAR
A WRITE command ~~with the CDB PARTITION_ID field containing a value that matches the contents of the credential PARTITION_ID field and the CDB USER_OBJECT_ID field containing a value that matches the contents of the object descriptor SINGLE OBJECT_ID field.~~	USER	WRITE	~~1OBJECT~~ U/C
Combinations of OBJECT TYPE field, PERMISSION BITS field, and OBJECT DESCRIPTOR TYPE field values not shown in this table and table 11 are reserved. The ~~credential and~~ capability fields not shown in this table may place additional limits on the objects that are allowed to be accessed.			
[a] ~~The object descriptor SINGLE OBJECT_ID field shall contain zero.~~			

Table 11 — ~~Capability fields that allow attribute~~ Attribute retrieval and setting functions allowed by specific capability field values (Sheet 1 of 5)

Attribute-Related Functions Allowed	Capability Field values that allow attribute-related functions		
	Object Type Name	Permission Bits That Are Set To One	Object Descriptor Name
Retrieval of attributes from the Current Command attributes page	USER or COLLECTION	none	U/C
Retrieval of attributes from the Current Command attributes page	PARTITION or ROOT	none	PAR
Retrieval of attributes from any attributes page associated with the addressed user object ~~The retrieval of attributes from any attributes page associated with the user object with a Partition_ID matching the value in the credential PARTITION_ID field and a User_Object_ID matching the value in the object descriptor SINGLE OBJECT_ID field.[a, b]~~	USER	GET_ATTR	~~1OBJECT~~ U/C
As part of a CREATE command or CREATE AND WRITE command, ~~the~~ retrieval of attributes from any attributes page associated with any user object created by the command.[a] ~~the user object with a Partition_ID matching the value in the credential PARTITION_ID field and the largest valued User_Object_ID created.[a]~~	USER	GET_ATTR	~~NONE~~ U/C
Retrieval of attributes from any attributes page associated with the addressed collection ~~The retrieval of attributes from any attributes page associated with the collection with a Partition_ID matching the value in the credential PARTITION_ID field and a Collection_Object_ID matching the value in the object descriptor SINGLE OBJECT_ID field.[a, b]~~	COLLECTION	GET_ATTR	~~1OBJECT~~ U/C
As part of a CREATE COLLECTION command, ~~the~~ retrieval of attributes from any attributes page associated with the collection created by the command ~~with a Partition_ID matching the value in the credential PARTITION_ID field and the Collection_Object_ID created.[a]~~	COLLECTION	GET_ATTR	~~NONE~~ U/C
Combinations of OBJECT TYPE field, PERMISSION BITS field, and OBJECT DESCRIPTOR TYPE field values not shown in this table and table 10 are reserved. The ~~credential and~~ capability fields not shown in this table may place additional limits on the objects that are allowed to be accessed.			
~~a~~ ~~Attributes in the Current Command attributes page may be retrieved even if the GET_ATTR bit is set to zero.~~ b ~~The command functions allowed for the GET ATTRIBUTES command and SET ATTRIBUTES command are specified by the GET_ATTR bit and SET_ATTR bit, respectively. The GET ATTRIBUTES command and SET ATTRIBUTES command are allowed if only the GET_ATTR bit, or SET_ATTR bit, or both are set to one.~~			

Table 11 — ~~Capability fields that allow attribute~~ Attribute retrieval and setting functions allowed by specific capability field values (Sheet 2 of 5)

Attribute-Related Functions Allowed	Capability Field values that allow attribute-related functions		
	Object Type Name	Permission Bits That Are Set To One	Object Descriptor Name
Retrieval of attributes from any attributes page associated with the addressed partition ~~The retrieval of attributes from any attributes page associated with the partition with a Partition_ID matching the value in the object descriptor SINGLE OBJECT_ID field.~~ ~~a, b~~	PARTITION	GET_ATTR	~~1OBJECT~~ PAR
As part of a CREATE PARTITION command, the retrieval of attributes from any attributes page associated with the created partition.~~ a~~	PARTITION	GET_ATTR	~~NONE~~ PAR
~~The retrieval~~ Retrieval of attributes from any attributes page associated with the root object or in any attributes page associated with partition zero (see 3.1.31).~~ a~~	ROOT	GET_ATTR	~~NONE or 1OBJECT~~ PAR
Setting attributes in any attributes page associated with the addressed user object, except attributes in a User Object Policy/Security attributes page ~~The setting of attributes in any attributes page, other than User Object Security attributes page, associated with the user object with a Partition_ID matching the value in the credential PARTITION_ID field and a User_Object_ID matching the value in the object descriptor SINGLE OBJECT_ID field.~~ ~~b~~	USER	SET_ATTR	~~1OBJECT~~ U/C
As part of a CREATE command or CREATE AND WRITE command, the setting of attributes in any attributes page associated with any user object created by the command, except attributes in a User Object Policy/Security attributes page ~~As part of a CREATE command or CREATE AND WRITE command, the setting of attributes in any attributes page, other than attributes in the User Object Policy/Security attributes page, associated with any user object created by the command the user object with a Partition_ID matching the value in the credential PARTITION_ID field and the largest valued User_Object_ID created.~~	USER	SET_ATTR	~~NONE~~ U/C

Combinations of OBJECT TYPE field, PERMISSION BITS field, and OBJECT DESCRIPTOR TYPE field values not shown in this table and table 10 are reserved.
The ~~credential and~~ capability fields not shown in this table may place additional limits on the objects that are allowed to be accessed.

~~a~~ ~~Attributes in the Current Command attributes page may be retrieved even if the GET_ATTR bit is set to zero.~~

b ~~The command functions allowed for the GET ATTRIBUTES command and SET ATTRIBUTES command are specified by the GET_ATTR bit and SET_ATTR bit, respectively. The GET ATTRIBUTES command and SET ATTRIBUTES command are allowed if only the GET_ATTR bit, or SET_ATTR bit, or both are set to one.~~

Table 11 — ~~Capability fields that allow attribute~~ Attribute retrieval and setting functions allowed by specific capability field values (Sheet 3 of 5)

Attribute-Related Functions Allowed	Capability Field values that allow attribute-related functions		
	Object Type Name	Permission Bits That Are Set To One	Object Descriptor Name
Setting attributes in any attributes page associated with the addressed collection, except attributes in a Collection Policy/Security attributes page ~~The setting of attributes in any attributes page, other than Collection Security attributes page, associated with the collection with a Partition_ID matching the value in the credential PARTITION_ID field and a Collection_Object_ID matching the value in the object descriptor SINGLE OBJECT_ID field.~~ [b]	COLLECTION	SET_ATTR	~~1OBJECT~~ U/C
As part of a CREATE COLLECTION command, the setting of attributes in any attributes page associated with the collection created by the command, except attributes in the Collection Policy/Security attributes page ~~As part of a CREATE COLLECTION command, the setting of attributes in any attributes page, other than attributes in the Collection Security attributes page, associated with the collection with a Partition_ID matching the value in the credential PARTITION_ID field and the Collection_Object_ID created.~~	COLLECTION	SET_ATTR	~~NONE~~ U/C
Setting attributes in any attributes page associated with the addressed partition, except attributes in a Partition Policy/Security attributes page ~~The setting of attributes in any attributes page, other than Partition Security attributes page, associated with the partition with a Partition_ID matching the value in the object descriptor SINGLE OBJECT_ID field.~~ [b]	PARTITION	SET_ATTR	~~1OBJECT~~ PAR
As part of a CREATE PARTITION command, the setting of attributes in any attributes page associated with the partition created by the command, except attributes in the Partition Policy/Security attributes page ~~As part of a CREATE PARTITION command, the setting of attributes in any attributes page other than Partition Security attributes page associated with the created partition.~~	PARTITION	SET_ATTR	~~NONE~~ PAR

Combinations of OBJECT TYPE field, PERMISSION BITS field, and OBJECT DESCRIPTOR TYPE field values not shown in this table and table 10 are reserved.

The ~~credential and~~ capability fields not shown in this table may place additional limits on the objects that are allowed to be accessed.

~~[a] Attributes in the Current Command attributes page may be retrieved even if the GET_ATTR bit is set to zero.~~

~~[b] The command functions allowed for the GET ATTRIBUTES command and SET ATTRIBUTES command are specified by the GET_ATTR bit and SET_ATTR bit, respectively. The GET ATTRIBUTES command and SET ATTRIBUTES command are allowed if only the GET_ATTR bit, or SET_ATTR bit, or both are set to one.~~

Table 11 — ~~Capability fields that allow attribute~~ Attribute retrieval and setting functions allowed by specific capability field values (Sheet 4 of 5)

Attribute-Related Functions Allowed	Capability Field values that allow attribute-related functions		
	Object Type Name	Permission Bits That Are Set To One	Object Descriptor Name
Setting attributes in any attributes page associated with the root object, except attributes in a Root Policy/Security attributes page Setting attributes in any attributes page associated with partition zero (see 3.1.31), except attributes in a Partition Policy/Security attributes page ~~The setting of attributes in any attributes page, other than Root Security attributes page, associated with the root object or in any attributes page, other than Partition Security attributes page, associated with partition zero (see 3.1.31).~~ [b]	ROOT	SET_ATTR	~~NONE or 1OBJECT~~ PAR
Setting attributes in any attributes page associated with the addressed user object ~~The setting of attributes in any attributes page associated with the user object with a Partition_ID matching the value in the credential PARTITION_ID field and a User_Object_ID matching the value in the object descriptor SINGLE OBJECT_ID field.~~ [b]	USER	SET_ATTR and ~~SECURITY~~ POL/SEC	~~1OBJECT~~ U/C
As part of a CREATE command or CREATE AND WRITE command, the setting of attributes in any attributes page associated with any user object created by the command ~~As part of a CREATE command or CREATE AND WRITE command, the setting of attributes in any attributes page associated the user object with a Partition_ID matching the value in the credential PARTITION_ID field and the largest valued User_Object_ID created.~~	USER	SET_ATTR and ~~SECURITY~~ POL/SEC	~~NONE~~ U/C
Setting attributes in any attributes page associated with the addressed collection ~~The setting of attributes in any attributes page associated with the collection with a Partition_ID matching the value in the credential PARTITION_ID field and a Collection_Object_ID matching the value in the object descriptor SINGLE OBJECT_ID field.~~ [b]	COLLECTION	SET_ATTR and ~~SECURITY~~ POL/SEC	~~1OBJECT~~ U/C
Combinations of OBJECT TYPE field, PERMISSION BITS field, and OBJECT DESCRIPTOR TYPE field values not shown in this table and table 10 are reserved. The ~~credential and~~ capability fields not shown in this table may place additional limits on the objects that are allowed to be accessed.			
~~a~~ ~~Attributes in the Current Command attributes page may be retrieved even if the GET_ATTR bit is set to zero.~~ b ~~The command functions allowed for the GET ATTRIBUTES command and SET ATTRIBUTES command are specified by the GET_ATTR bit and SET_ATTR bit, respectively. The GET ATTRIBUTES command and SET ATTRIBUTES command are allowed if only the GET_ATTR bit, or SET_ATTR bit, or both are set to one.~~			

Table 11 — ~~Capability fields that allow attribute~~ Attribute retrieval and setting functions allowed by specific capability field values (Sheet 5 of 5)

Attribute-Related Functions Allowed	Capability Field values that allow attribute-related functions		
	Object Type Name	Permission Bits That Are Set To One	Object Descriptor Name
As part of a CREATE COLLECTION command, the setting of attributes in any attributes page associated with the collection created by the command ~~As part of a CREATE COLLECTION command, the setting of attributes in any attributes page associated with the collection with a Partition_ID matching the value in the credential PARTITION_ID field and the Collection_Object_ID created.~~	COLLECTION	SET_ATTR and ~~SECURITY~~ POL/SEC	~~NONE~~ U/C
Setting attributes in any attributes page associated with the addressed partition ~~The setting of attributes in any attributes page associated with the partition with a Partition_ID matching the value in the object descriptor SINGLE OBJECT_ID field.~~ [~~b~~]	PARTITION	SET_ATTR and ~~SECURITY~~ POL/SEC	~~1OBJECT~~ PAR
As part of a CREATE PARTITION command, the setting of attributes in any attributes page associated with the partition created by the command ~~As part of a CREATE PARTITION command, the setting of attributes in any attributes page associated with the created partition.~~	PARTITION	SET_ATTR and ~~SECURITY~~ POL/SEC	~~NONE~~ PAR
~~The setting of~~ Setting attributes in any attributes page associated with the root object or in any attributes page associated with partition zero (see 3.1.31).~~—~~ [~~b~~]	ROOT	SET_ATTR and ~~SECURITY~~ POL/SEC	~~NONE or~~ ~~1OBJECT~~ PAR

Combinations of OBJECT TYPE field, PERMISSION BITS field, and OBJECT DESCRIPTOR TYPE field values not shown in this table and table 10 are reserved.
The ~~credential and~~ capability fields not shown in this table may place additional limits on the objects that are allowed to be accessed.

[~~a~~] ~~Attributes in the Current Command attributes page may be retrieved even if the GET_ATTR bit is set to zero.~~

b ~~The command functions allowed for the GET ATTRIBUTES command and SET ATTRIBUTES command are specified by the GET_ATTR bit and SET_ATTR bit, respectively. The GET ATTRIBUTES command and SET ATTRIBUTES command are allowed if only the GET_ATTR bit, or SET_ATTR bit, or both are set to one.~~

4.x.3 Policy access tags

The policy access tag (see table 12) allows the coordinated actions of both the OSD logical unit and policy/storage manager to prevent unsafe or temporarily undesirable utilization of OBSD storage that is assigned to the OSD logical unit.

Table 12 — Policy access tag format

Bit / Byte	7	6	5	4	3	2	1	0
0	FENCE	(MSB)						
1				VERSION				
2								
3								(LSB)

During normal operation the value of the FENCE bit is zero.

If the OSD logical unit detects a condition that would make further accesses to one or more OSD objects unsafe, it shall set the FENCE bit to one in the policy access tag attributes in the Policy/Security attributes pages associated with those objects (e.g., the User Object Policy/Security attributes page (see 7.1.2.23) if the OSD object is a user object) and notify the policy/storage manager of a condition needing attention. The OSD logical unit, policy/storage manager, or both act to correct whatever conditions are making accesses to the OSD objects unsafe. After the conditions making accesses to the OSD objects unsafe are corrected the policy/storage manager sets the FENCE bit to zero.

If a set attributes list (see 5.2.1.3) contains a request to set the FENCE bit to one, 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 PARAMETER LIST. If the CDB SET ATTRIBUTE NUMBER field contains 4000 0001h (i.e., the security version policy access tag attribute) and the set attributes data specified by the SET ATTRIBUTES OFFSET field (see 5.2.1.2) specifies that the FENCE bit be set to one, 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.

To block capability-based access to one or more OSD objects, the policy/storage manager changes the VERSION field in the policy access tag attributes in the Policy/Security attributes pages associated with those objects. The conditions under which the policy/storage manager may be call on to do this include:

a) Recovery from errors other than those detected by the OSD logical unit that make accesses to one or more OSD object unsafe; and
b) Receipt of a request to change the policy access tag from the security manager (see 4.9.5.5).

If a set attributes list (see 5.2.1.3) contains a request to set the VERSION field to zero, 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 PARAMETER LIST. If the CDB SET ATTRIBUTE NUMBER field contains 4000 0001h (i.e., the security version policy access tag attribute) and the set attributes data specified by the SET ATTRIBUTES OFFSET field (see 5.2.1.2) specifies that the VERSION field be set to zero, 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 OSD logical unit shall not modify the contents of a policy access tag VERSION field.

The device server terminates any command received with a capability whose POLICY ACCESS TAG field contains a non-zero value that differs from the policy access tag attribute value in the Policy/Security attributes page associated with the objects (see 4.x.2.2).

4.9 Security

4.9.1 Basic security model

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

a) An OBSD (see 3.1.26);
b) A policy/storage manager (see 4.x);
c) A security manager; and
d) Application clients.

~~Because the OSD security model is a credential-based access control system, all requests to the OSD logical unit should be accompanied by a valid credential that allows the application client to perform the requested command functions.~~ The principal function of the security manager is preparing credentials in response to application client requests. A credential is a data structure ~~prepared by the security manager (see 3.1.38)~~ containing a capability prepared by the policy/storage manager and protected by an integrity check value (see 3.1.18), having the following properties:

a) The capability in the credential grants defined access to an OSD logical unit for specific command functions (see 3.1.10); and
b) The integrity check value in the credential protects the capability and commands that include the capability from various attacks described in 4.9

~~that is sent to an application client in order to grant defined access to an OSD logical unit for specific command functions (see 3.1.10) performed on specific OSD objects. The credential includes a capability (see 3.1.4) that the application client copies to each CDB that requests the specified command functions.~~ ~~a cryptographically secured capability and a capability is a set of access rights allowed to the holder of the credential on an OSD object or set of OSD objects.~~

Figure 2 shows the flow of transactions between the components of the OSD security model.

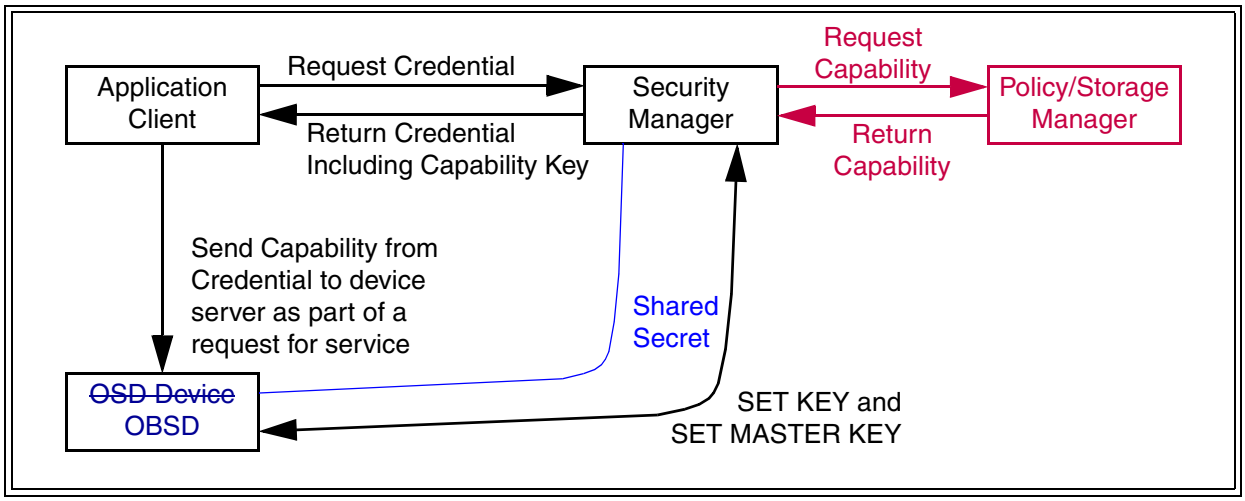

Figure 2 — OSD security model transactions

The security manager generates credentials, including capabilities prepared by the policy/storage manager, for authorized application clients at the request of an application client. The security manager returns a capability key with each credential. The credential gives the application client access to specific OSD components. The capability key allows the application client and device server to authenticate the commands and data they exchange with an integrity check value (see 4.9.7).

The protocol between the application client and the security manager is not defined by this standard~~;~~. ~~h~~However, the structure of the credential returned from the security manager to the application client is.

~~The~~ If any security method except NOSEC is used, the device server validates each command received from an application client to confirm that:

a) The credential has not been tampered with (i.e., that the credential was generated by the security manager and includes an integrity check value using a secret key known only to the security manager and OSD device server); and
b) The credential was rightfully obtained by the application client from the security manager 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 capability key to provide a proper integrity check value or values for the command).~~; and~~
~~c) The requested command function is permitted by the capability in the credential based on:~~
~~A) The type of functions (e.g., read, write, attributes setting, attributes retrieval); and~~
~~B) The OSD object on which the command functions are to be performed.~~

The capability key allows the OSD device server to validate that an application client rightfully obtained a credential and that the capability has not been tampered with. An application client that has just the capability (e.g., obtained by monitoring CDBs sent to the OSD device server) but not the capability key is unable to generate commands with valid integrity check value, meaning that application client is denied access to the OSD logical unit. This protocol ~~does~~ allows delegation of a credential if a application client delegates both the credential and the capability key.

The application client requests credentials and capability keys from the security manager for the command functions it needs to perform and sends those capabilities in those credentials to the OSD device server as part of commands that include an integrity check value using the capability key. While the application client is not trusted to follow this protocol, an application client that does not follow the protocol is unlikely to receive service from the OSD device server. ~~While the application client is not trusted to follow this protocol, the protocol is structured in a way that makes it in the application client's self-interest to follow the protocol. An application client that does not follow the protocol is unlikely to receive service from the OSD device server.~~

The security manager may authenticate the application client, but the OSD device server does not ~~need to~~ authenticate the application client. It is sufficient for the OSD device server to verify the capabilities and integrity check values ~~send~~ sent by the application client.

The security manager maintains security policy information the definition of which is outside the scope of this standard.

4.9.2 Trust assumptions

This subclause describes how each component of the OSD security model trusts the other components.

The OBSD is a trusted component, meaning that once an application client authenticates that it is communicating with a specific OSD logical unit using methods outside the scope of this standard, it trusts the OBSD 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.

The security manager is a trusted component. After the security manager is authenticated by the application client and by the OBSD using methods outside the scope of this standard, the security manager is trusted to:

a) Safely store long-lived keys;
b) Apply access controls correctly according to requirements that are outside the scope of this standard;
c) Perform the security functions defined for it by this standard; and
d) Not be controlled in a way that operates to the detriment of the application client's or OSD logical unit's interests.

The application client is not a trusted component. However, the OSD security model is defined so that the application client receives service from the OSD device server only if it interacts with both the security manager and the OSD device server in ways that assure the propriety of the application client's actions. Application clients should protect capability keys from disclosure to unauthorized entities.

The OSD security model components are trusted to maintain some degree of synchronization between their clocks. The OSD security model includes features designed to minimize dependencies on the degree of clock synchronization maintained by application clients (see 4.9.6).

Regardless of where the security manager resides (see 4.4), communications between the security manager and other components are trusted based on the requirements shown in table 13.

Table 13 — Security manager communications trust requirements

Component	Security Manager communications trust requirement
OSD device server	None
Application client	Confidential [a]
Policy/storage manager	None
[a] Confidential communications shall be protected from eavesdropping by physical or cryptographic means.	

~~When sending credentials to an application client, the security manager shall use a private, authenticated communications mechanism. The security manager may reside in the OBSD, in applications clients, or as a separate entity, but the security requirements on the communications mechanism shall not change based on the location of the security manager.~~

4.9.m Preparing credentials

In response to a request from an application client, the security manager shall prepare and return a credential as follows:

1) Forward the access requests from the application client to the policy/storage manager. If the policy/storage manager denies the forwarded request an error shall be returned to the requesting application client;
2) Insert the capability returned by the policy/storage manager in the credential;
3) Set the credential OSD SYSTEM ID field to the value in the OSD system ID attribute in the Root Information attributes page (see 7.1.2.8) of the OSD logical unit to which the credential applies;
4) Set the capability SECURITY METHOD field as follows:
   A) Select a security method other than the partition default:
      a) If the application client requested use of a specific security method, and use of the requested security method is allowed by both the addressed partition and the maintained security policy information, set the capability SECURITY METHOD field to the requested value;

b) If the maintained security policy information requires use of a specific security method for the requesting application client, set the capability SECURITY METHOD field to that value;

or

B) Use the partition default:

a) If the application client requested a credential to be used the SET KEY command (see 6.23) or the SET MASTER KEY command (see 6.24), set the capability SECURITY METHOD field to the value in the default security method attribute in the Root Policy/Security attributes page (see 7.1.2.20);

b) Otherwise, set the capability SECURITY METHOD field to the value in the default security method attribute in the Partition Policy/Security attributes page (see 7.1.2.21) for the partition whose Partition ID is contained in the capability ALLOWED PARTITION_ID field;

5) If the SECURITY METHOD field contains NOSEC, place zero in the CREDENTIAL INTEGRITY CHECK VALUE field and return the credential to the application client;

Otherwise:

6) Set the capability KEY VERSION field to the number of the working key ~~contains a value that the device server uses to identify the~~ secret key ~~to be~~ used ~~in computing~~ to compute the credential integrity check value. If a secret key other than a working key is used to compute the credential integrity check value (e.g. for a~~. For capabilities used with the~~ SET KEY command (see 6.23) or ~~to update secret keys other than working keys and~~ SET MASTER KEY command (see 6.24)), then set the capability KEY VERSION field to zero ~~the key version shall be zero.~~;

7) Set the capability INTEGRITY CHECK VALUE ALGORITHM field to the value that specifies the algorithm used to compute all integrity check values related to this credential ~~command~~. The algorithm ~~may be~~ shall be one of those identified ~~using~~ by the supported integrity check value algorithm attributes in the Root Policy/Security attributes page (see 7.1.2.20) and the INTEGRITY CHECK VALUE ALGORITHM field shall be set as described in 7.1.2.20~~ The value in the Root Security attribute whose attribute number equals the contents of the INTEGRITY CHECK VALUE ALGORITHM field plus 8000 0000h identifies the integrity check value algorithm~~;

8) As specified by the maintained security policy information, modify other capability fields, including but not limited to the following:

A) Setting the CAPABILITY EXPIRATION TIME field to a value that is consistent with the secret key management policy;

B) Ensuring that the capability AUDIT field and CAPABILITY DISCRIMINATOR field contain non-zero values;

C) Setting the capability OBJECT CREATED TIME field to a non-zero value as described in 4.x.2.2.1; and

D) Ensuring that the POL/SEC bit in the PERMISSIONS BIT MASK field is set to zero, if appropriate;

9) Compute the credential integrity check value as described in 4.9.5.4 and place the result in the CREDENTIAL INTEGRITY CHECK VALUE field in the credential;

and

10) Return the credential thus constructed to the application client with the credential integrity check value serving as the capability key.

Successful use of the capability expiration time (see step A) in step 8)) requires some degree of synchronization between the clocks of the device server and security manager. The protocol for synchronizing the clocks is outside the scope of this standard, however, the protocol should be implemented in a secure manner (i.e., it should not be possible for an adversary to set the clock in the device server backwards to enable the replay of expired credentials).

4.9.3 Security methods

4.9.3.1 Introduction

This standard defines several security methods (see table 14).

Table 14 — OSD security methods

Security Method	Description	Reference
NOSEC	No security	4.9.3.2
CAPKEY	Integrity of capabilities	4.9.3.3
CMDRSP	Integrity of CDB, status, and sense data	4.9.3.4
ALLDATA	Integrity of all data in transit	4.9.3.5

~~The security method used by one partition may be different from the security method used by another partition.~~

~~A command prepared for a security mode other than the one the device server uses for processing may complete without errors (e.g., a command prepared for the ALLDATA security method may complete without errors reported by the device server if the CMDRSP security method is in use because the preparations for the ALLDATA security method include the preparations that are necessary for the CMDRSP security method).~~

Editors Note 1 - ROW: The discussion of secure channel features includes the following text: “Data Confidentiality (privacy) Optional: if message is read, it cannot be understood other than by the unauthorized parties”. This text is not included in the table below. Either Data Confidentiality is required to satisfy the threat analysis shown in the table, meaning that Data Confidentiality is required, not optional, or Data Confidentiality is not required, meaning that its absence from the table footnote is appropriate.

The OSD security methods are designed to address zero or more specific security threats (see table 15).

Table 15 — Security methods and threats thwarted

Threat	Threat thwarted by security method				
		CAPKEY			
		Over secure channel [a]			
	NOSEC	No	Yes	CMDRSP	ALLDATA
Forgery of credential	No	Yes	Yes	Yes	Yes
Alteration of capabilities	No	Yes	Yes	Yes	Yes
Unauthorized use of credential	No	No	Yes	Yes	Yes
Replay of command or status	No	No	No	Yes	Yes
Alteration of command or status	No	No	No	Yes	Yes
Replay of data	No	No	No	No	Yes
Alteration of data	No	No	No	No	Yes
Inspection of command, status or data	No	No	No	No	No

[a] A secure channel provides the following security guarantees:
a) Cryptographic integrity: Any message received is the one was sent (i.e., no tampering occurred). Messages in which tampering is detected are discarded;
b) Data origin authentication: The message received originated from the authenticated originator within the limits of the secure channel authentication mechanism;
c) Replay protection: The same message is not delivered multiple times and that there is a limited number of out-of-order messages.

4.9.3.2 The NOSEC security method

In the NOSEC security method, no OSD security features or algorithms are used by the device server. If the root object and all partitions in the OSD logical unit use the NOSEC security method, then:

a) Specific SPC-3 commands (e.g., LOG SENSE) may be sent (see table 41 in 6.1) without encapsulating them in the PERFORM SCSI COMMAND command ~~(see 6.11)~~ (see 6.16); and
b) Persistent reservations (see 4.15) are allowed for the logical unit.

4.9.3.3 The CAPKEY security method

The CAPKEY security method validates the integrity of the capability information in each CDB.

The application client computes the CDB REQUEST INTEGRITY CHECK VALUE field (see 5.2.6) contents using:

a) The algorithm specified in the capability INTEGRITY CHECK VALUE ALGORITHM field (see 4.x.2.2);
b) ~~A~~ The security token returned in the Security Token VPD page (see 7.5.3) ~~described in this subclause~~; and
c) The credential capability key (see 4.9.4.2).

The device server validates the credential ~~and capabilities~~ as described in 4.9.5.2.

~~The security token used in the request integrity check value computation and validation shall be the concatenation of the following two values, defined in SAM-3 and the applicable SCSI transport protocol:~~

~~1) Initiator identifier; and~~
~~2) Target identifier.~~

~~The initiator identifier and target identifier shall be those for the I_T nexus over which the command request is made and responded to. Both the application client and the device server shall concatenate the two values in the order shown in this subclause.~~

~~If the SCSI transport protocol over which the I_T Nexus is formed is capable of changing the initiator identifier and target identifier in an I_T Nexus without generating an I_T Nexus Loss event notification (see SAM-3), the security token shall be recomputed for every command sent.~~

~~If the application client and device server are in separate SCSI domains (e.g., connected by a bridging device, see SAM-3) the security tokens built by the application client and device server have a high probability of being different, resulting in comparison failures of the request integrity check values. The credential should appear to have been tampered with when this is not truly the case. The problems caused by multiple SCSI domains may be solved by using a different security method (e.g., the CMDRSP security method).~~

The CAPKEY security method is useful when the service delivery subsystem between the OSD device server and application client is secured via methods specified in the applicable SCSI transport protocol. Combining the CAPKEY security method with a secure transport (i.e., one that provides an authenticated channel) provides the same protections as the ALLDATA security method. Even when communications are secured by such means, it is necessary to prevent the untrusted application client (see 4.9.2) from forging or ~~otherwise~~ modifying a credential, and from replaying a credential on a different secure channel. ~~credential.~~

4.9.3.4 The CMDRSP security method

The CMDRSP security method validates the integrity of the CDB, status, and sense data for each command.

The application client computes the CDB REQUEST INTEGRITY CHECK VALUE field (see 5.2.6) contents using:

a) The algorithm specified in the capability INTEGRITY CHECK VALUE ALGORITHM field (see 4.x.2.2);
b) All the bytes in the CDB with the bytes in the REQUEST INTEGRITY CHECK VALUE field set to zero; and
c) The credential capability key (see 4.9.4.2).

The device server validates the credential ~~and capabilities~~ as described in 4.9.5.2.

If the credential ~~and capabilities~~ validation process successfully validates the integrity check value associated with the command, the device server shall:

1) Compute an integrity check value for the response data using:
   A) The algorithm specified in the capability INTEGRITY CHECK VALUE ALGORITHM field (see 4.x.2.2);
   B) The following array of bytes:
      1) The request nonce from the CDB (see 5.2.6);
      2) The status byte; and
      3) If the status is CHECK CONDITION, the sense data with the RESPONSE INTEGRITY CHECK VALUE field in the OSD response integrity check value sense data descriptor (see 4.13.x.y) ~~(see SPC-3)~~ set to zero;

      and
   C) The capability key (see 4.9.4.2) for the reconstructed credential (see 4.9.5.3);

   and
2) Place the computed integrity check value in the following location:
   A) If the status is not CHECK CONDITION, the computed integrity check value shall be placed in the response integrity check value attribute in the Current Command attributes page (see 7.1.2.24); or
   B) If the status is CHECK CONDITION, the computed integrity check value shall be placed in the RESPONSE INTEGRITY CHECK VALUE field in the OSD response integrity check value sense data descriptor (see 4.13.x.y) in the sense data.

If the credential ~~and capabilities~~ validation process fails to validate the integrity check value associated with the command, the device server shall place zero in the RESPONSE INTEGRITY CHECK VALUE field in the OSD response integrity check value sense data descriptor in the sense data.

If the status is not CHECK CONDITION, the application client validates the response integrity check value by recomputing it as described in this subclause and comparing the result to the value of the response integrity check value attribute in the Current Command attributes page.

If the status is CHECK CONDITION, the application client validates the response integrity check value by:

1) Saving the response integrity check value found in the RESPONSE INTEGRITY CHECK VALUE field in the OSD response integrity check value sense data descriptor in the sense data;
2) Placing zero in the response integrity check value found in the RESPONSE INTEGRITY CHECK VALUE field in the OSD response integrity check value sense data descriptor;
3) Recomputing the response integrity check value as described in this subclause; and
4) Comparing the result to the value saved in step 1).

If the application client fails in validating the response integrity check value as described in this subclause, it should take a recovery action not specified by this standard~~. One~~ (e.g., one possible action is to request a new credential from the security manager and retry the command). If the error reoccurs, alternate recovery actions should be considered and the presence of malicious entities perpetrating a denial of service attack should be considered.

The CMDRSP security method may be used when the service delivery subsystem between the OSD device server and application client is not secured. The CMDRSP security method protects against corruption of the command, command parameter data, status, and sense data ~~and CDB parameters~~ while avoiding the overhead that may be required to protect all transferred data. Use of the CMDRSP security method prevents an untrusted application client from forging, modifying or replaying a capability.

4.9.3.5 The ALLDATA security method

The ALLDATA security method validates the integrity of all data in transit between an application client and device server.

The application client computes the CDB REQUEST INTEGRITY CHECK VALUE field (see 5.2.6) contents using the same algorithm specified for the CMDRSP security method (see 4.9.3.4). The device server validates the credential ~~and capabilities~~ as described in 4.9.5.2.

The application client also computes the data-out integrity check value using:

a) The algorithm specified in the capability INTEGRITY CHECK VALUE ALGORITHM field (see 4.x.2.2);
b) The used bytes in the following Data-Out Buffer segments (see 4.11.3)~~;~~:
   1) Command data or parameter data;
   2) Set attributes; and
   3) Get attributes;

   and
c) The credential capability key (see 4.9.4.2).

The application client places the data-out integrity information (see table 16) in the Data-Out Buffer starting at the byte specified by the CDB DATA-OUT INTEGRITY CHECK VALUE OFFSET field (see 5.2.6).

Table 16 — Data-out integrity information format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			NUMBER OF COMMAND OR PARAMETER BYTES				
7								(LSB)
8	(MSB)			NUMBER OF SET ATTRIBUTES BYTES				
15								(LSB)
16	(MSB)			NUMBER OF GET ATTRIBUTES BYTES				
23								(LSB)
24	(MSB)			DATA-OUT INTEGRITY CHECK VALUE				
43								(LSB)

The NUMBER OF COMMAND OR PARAMETER BYTES field specifies the number of bytes from the command data or parameter data segment that are included in the data-out integrity check value. If the value in the CDB LENGTH field, if any, or the value in the CDB PARAMETER LIST LENGTH field, if any, is larger than the value in the NUMBER OF COMMAND OR PARAMETER BYTES field, 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 NUMBER OF SET ATTRIBUTES BYTES field specifies the number of bytes from the set attributes segment that are included in the data-out integrity check value. If the value in the CDB SET ATTRIBUTE LENGTH field, if any, or the value in the CDB SET ATTRIBUTES LIST LENGTH field, if any, is larger than the value in the NUMBER OF SET ATTRIBUTES BYTES field, 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 NUMBER OF GET ATTRIBUTES BYTES field specifies the number of bytes from the get attributes segment that are included in the data-out integrity check value. If the value in the CDB GET ATTRIBUTES LIST LENGTH field, if any, is larger than the value in the NUMBER OF GET ATTRIBUTES BYTES field, 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 DATA-OUT INTEGRITY CHECK VALUE field contains the data-out integrity check value computed by the application client.

The device server shall validate the data-out integrity check value by:

1) Computing an integrity check value using:
   A) The algorithm specified in the capability INTEGRITY CHECK VALUE ALGORITHM field;
   B) The following bytes from Data-Out Buffer~~;~~:
      1) The number of bytes specified by the NUMBER OF COMMAND OR PARAMETER BYTES field starting at the Data-Out Buffer byte offset zero;
      2) The number of bytes specified by the NUMBER OF SET ATTRIBUTES BYTES field starting at the Data-Out Buffer byte offset specified by the CDB SET ATTRIBUTES LIST OFFSET field (see 5.2.1.3); and
      3) The number of bytes specified by the NUMBER OF GET ATTRIBUTES BYTES field starting at the Data-Out Buffer byte offset specified by the CDB GET ATTRIBUTES LIST OFFSET field (see 5.2.1.3);

      and
   C) The capability key (see 4.9.4.2) for the reconstructed credential (see 4.9.5.3);

   and
2) Comparing the results to contents of the DATA-OUT INTEGRITY CHECK VALUE field.

If the validation fails, the state of the OSD objects and attributes shall not be altered in any detectable way, 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 DATA-OUT BUFFER INTEGRITY CHECK VALUE.

The device server shall compute the response integrity check value using the same algorithm specified for the CMDRSP security method (see 4.9.3.4) and the application client validates the response integrity check value using the same algorithm specified for the CMDRSP security method.

The device server shall compute the data-in integrity check value using:

a) The algorithm specified in the capability INTEGRITY CHECK VALUE ALGORITHM field;
b) The used bytes in the following Data-In Buffer segments (see 4.11.2)~~;~~:
   1) Command data or parameter data; and
   2) Retrieved attributes;

   and
c) The capability key (see 4.9.4.2) for the reconstructed credential (see 4.9.5.3).

The device server shall place the data-in integrity information (see table 17) in the Data-In Buffer starting at the byte specified by the CDB DATA-IN INTEGRITY CHECK VALUE OFFSET field (see 5.2.6).

Table 17 — Data-in integrity information format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			NUMBER OF COMMAND OR PARAMETER BYTES				
7								(LSB)
8	(MSB)			NUMBER OF RETRIEVED ATTRIBUTES BYTES				
15								(LSB)
16	(MSB)			DATA-IN INTEGRITY CHECK VALUE				
35								(LSB)

The NUMBER OF COMMAND OR PARAMETER BYTES field specifies the number of bytes from the command data or parameter data segment that are included in the data-in integrity check value.

The NUMBER OF RETRIEVED ATTRIBUTES BYTES field specifies the number of bytes from the retrieved attributes segment that are included in the data-in integrity check value.

The DATA-IN INTEGRITY CHECK VALUE field contains the data-in integrity check value computed by the device server.

After status has been received, the application client validates the data-in integrity check value by:

1) Computing an integrity check value using:
   A) The algorithm specified in the capability INTEGRITY CHECK VALUE ALGORITHM field;
   B) The following bytes from Data-In Buffer~~;~~:
      1) The number of bytes specified by the NUMBER OF COMMAND OR PARAMETER BYTES field starting at the Data-In Buffer byte offset zero; and
      2) The number of bytes specified by the NUMBER OF RETRIEVED ATTRIBUTES BYTES field starting at the Data-In Buffer byte offset specified by the CDB RETRIEVED ATTRIBUTES OFFSET field (see 5.2.1);
      and
   C) The credential capability key (see 4.9.4.2);
   and
2) Comparing the results to contents of the DATA-IN INTEGRITY CHECK VALUE field.

If the application client fails in validating the data-in integrity check value, it should take a recovery action not specified by this standard~~. One~~ (e.g., one possible action is to request a new credential from the security manager and retry the command). If the error reoccurs, alternate recovery actions should be considered and the presence of malicious entities perpetrating a denial of service attack should be considered.

The ALLDATA security method provides for applying integrity check values to every byte exchanged between the application client and OSD device server. Protection is provided against network attacks similar to those protected against by the security architecture for the internet protocol when confidentiality is not used (see RFC 2401), at the expense of computing and validating numerous integrity check values.

4.9.4 Credentials ~~and capabilities~~

4.9.4.1 Credential format

A credential (see table 18) is transferred from the security manager to an application client over a communications mechanism that meets the requirements specified in 4.9.2. ~~encrypts the data it transfers for data privacy.~~

Table 18 — Credential format

Bit / Byte	7	6	5	4	3	2	1	0
0	Capability (see 4.x.2.2)							
79								
80	OSD SYSTEM ID							
99								
100	(MSB) CREDENTIAL INTEGRITY CHECK VALUE							
119								(LSB)

The capability is described in 4.x.2.2.

The OSD SYSTEM ID field specifies the value in the OSD system ID attribute in the Root Information attributes page (see 7.1.2.8) of the OSD logical unit to which the credential applies.

The CREDENTIAL INTEGRITY CHECK VALUE field contains an integrity check value (see 4.9.7) that is computed using the algorithm, inputs, and secret key specified in 4.9.5.4.

4.9.4.2 Capability key

All security methods except the NOSEC security method require the computation of one or more integrity check values using a capability key as the secret key (see 3.1.37).

For application clients, the capability key is the contents of the CREDENTIAL INTEGRITY CHECK VALUE field (see 4.9.4.1).

The device server processing of each command relies on only the capability portion of the credential (see 4.9.4.1) that the application client has copied into the CDB. Since the capability does not include the CREDENTIAL INTEGRITY CHECK VALUE field, the device server needs to compute the capability key for each processed command ~~as follows~~ by:

1) Reconstructing the credential containing the CDB capability as described in 4.9.5.3; and
2) Computing the credential integrity check value for the reconstructed credential using the algorithm, inputs, and secret key specified in 4.9.5.4.

NOTE 1 The two steps used by the device server to compute capability key are the first two steps that the device server uses to validate the capability contained in the CDB (see 4.9.5.2). The device server may perform these two steps ~~only~~ once for every command processed.

4.9.5 OSD device server security algorithms

~~#### 4.9.5.1 Determining the security method to use for processing a command~~

~~The device server shall process the SET KEY command (see 6.23) and the SET MASTER KEY command (see 6.24) using the security method specified by the security method attribute in the Root Policy/Security attributes page (see 7.1.2.20).~~

~~The device server shall process all other commands using the security method specified by the security method attribute in the Partition Policy/Security attributes page (see 7.1.2.21) of the partition specified by the PARTITION_ID field in the CDB. If the CDB does not contain a PARTITION_ID field, the command shall be processed using the security method specified by the security method attribute in the Partition Policy/Security attributes page for partition zero.~~

4.9.5.2 Credential ~~and capability~~ validation

Editors Note 2 - ROW: The exceptions described in the following paragraph are covered as described in the editor's notes in 4.9.8.2.

~~This subclause describes the process for validating a capability received in a CDB if the security method is use is not NOSEC. With two exceptions, the process described in this subclause apply to the validation of all commands. Additional validation requirements that apply only to the SET KEY command (see 6.23) and SET MASTER KEY command (see 6.24) are specified in 4.9.8.2.~~

~~The algorithm for determining which security method to use is specified in 4.9.5.1.~~

The processes described in this subclause do not apply if the CDB SECURITY METHOD field specifies the NOSEC security method (i.e., if the CDB SECURITY METHOD field contains zero).

~~If the security method is use is not NOSEC and a command with zero in the CREDENTIAL/CAPABILITY FORMAT field (see 4.x.2.2) is received, 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.~~

Editors Note 3 - ROW: The next two paragraphs are being removed because reflector discussions suggest that the credential validation order is strict. The capability verification order is not strict, but that is in 4.x.2.2 now.

~~The order in which validation actions are described in this subclause follows the order in which the fields being validated appear in the capability (see 4.x.2.2) and credential (see 4.9.4.1). In most cases the order in which the validation steps are performed is not critical, but the validation of the request nonce should be performed as early in the process as possible to increase the chances of detecting and rejecting duplicate nonces.~~

~~It may be possible to optimize the process described in this subclause, but such optimizations are beyond the scope of this standard. In addition to the conditions described in this subclause, the device server may return CHECK CONDITION status for any anomalies it detects.~~

The device server shall validate~~s~~ the credential associated with a CDB ~~the capability information in a CDB~~ by:

1) Reconstructing the credential containing the capability as described in 4.9.5.3;
2) Computing the credential integrity check value for the reconstructed credential using the algorithm, inputs, and secret key specified in 4.9.5.4;
3) Computing the request integrity check value using:
   A) The algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM ~~FIELD~~ field in the capability;
   B) ~~One~~ Based on the contents of the CDB SECURITY METHOD field, one of the following arrays of bytes:
      a) For the CAPKEY security method, the security token ~~(see 4.9.3.2)~~ (see 4.9.3.2); or
      b) For the CMDRSP security method and the ALLDATA security method, all the bytes in the CDB with the bytes in the REQUEST INTEGRITY CHECK VALUE field set to zero;

      and
   C) The credential integrity check value computed in step 2) as the secret key;

   ~~and~~
4) Verifying that the request integrity check value matches the contents of the CDB REQUEST INTEGRITY CHECK VALUE field ~~in the CDB~~ (see 5.2.6). If the contents in the request integrity check value field in the CDB do not match the computed integrity check value, 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; and
5) If the CDB SECURITY METHOD field specifies the CMDRSP security method or the ALLDATA security method, validate the CDB REQUEST NONCE field as described in 4.9.6.2.

If the validation of a credential ~~and capability~~ results in a CHECK CONDITION status being returned, the state of the OSD objects and attributes shall not be altered ~~in any detectable way~~.

4.9.5.3 Reconstructing the credential

The device server reconstructs a credential from a CDB capability by:

1) ~~Reconstructing the credential fields~~ Copying the value in the OSD system ID attribute in the Root Information attributes page (see 7.1.2.8) to the OSD SYSTEM ID field of the reconstructed credential; and ~~as follows:~~
   ~~A) Copy the value in the OSD system ID attribute in the Root Information attributes page (see 7.1.2.8) to the OSD SYSTEM ID field of the reconstructed credential;~~
   ~~and~~

~~B) Reconstruct the credential PARTITION_ID field as follows:~~
~~a) If the command is CREATE PARTITION, FLUSH PARTITION, FLUSH OSD, FORMAT OSD, LIST, PERFORM SCSI COMMAND, REMOVE PARTITION, or SET KEY, then place zero in PARTITION_ID field of the reconstructed credential;~~
~~b) If the CDB USER_OBJECT_ID field contains zero and the command is FLUSH OBJECT or PERFORM TASK MANAGEMENT, then place zero in PARTITION_ID field of the reconstructed credential; or~~
~~c) Otherwise, copy the contents of the CDB PARTITION_ID field to the PARTITION_ID field of the reconstructed credential;~~
~~and~~

2) Copying the capability from the CDB to the reconstructed credential.

The CREDENTIAL INTEGRITY CHECK VALUE field is not used in a reconstructed credential.

4.9.5.4 Computing the credential integrity check value

The credential integrity check value shall be computed using:

a) The algorithm specified by the INTEGRITY CHECK VALUE ALGORITHM FIELD in the capability;
b) The following bytes:
   A) All of the bytes in all of the fields defined for the credential (see 4.9.4.1);
   B) Except the bytes in the CREDENTIAL INTEGRITY CHECK VALUE field;
   and
c) The secret key selected as follows;
   A) If the OBJECT TYPE field in the capability (see 4.x.2.2) contains COLLECTION or USER, the secret key is the authentication working key:
      a) Identified by the KEY VERSION field in the capability; and ~~for~~
      b) Associated with the partition identified by the PARTITION_ID field in the CDB;
   B) If the OBJECT TYPE field in the capability contains ROOT or PARTITION and the command is not SET KEY and not SET MASTER KEY the secret key is, the authentication working key for partition zero identified by the KEY VERSION field in the capability;
   C) If the command is SET KEY, the secret key ~~that~~ is selected as follows:
      a) If the KEY TO SET field in the CDB (see 6.23) contains 01b (i.e., update ~~drive~~ root key), the authentication master key;
      b) If the KEY TO SET field in the CDB contains 10b (i.e., update partition key), the authentication ~~drive~~ root key; or
      c) If the KEY TO SET field in the CDB contains 11b (i.e., update working key), the authentication partition key for the partition identified by the PARTITION_ID field in the CDB;
      or
   D) For the SET MASTER KEY command:
      a) For the seed exchange step (see 6.20.2), the authentication master key; or
      b) For the change master key step (see 6.20.3), the next authentication master key computed after GOOD status has been returned by the set master key seed step.

4.9.5.5 Invalidating credentials

The security manager may invalidate the credentials for one OSD object by ~~changing~~ requesting that the policy/storage manager change the ~~security version~~ policy access tag ~~key~~ attribute in the policy/security attributes page associated with that OSD object (see 4.x) ~~(see table 8 in 4.x.2.2)~~ to a value other than the ~~security version~~ policy access tag ~~key~~ value that is contained in the credential's capability. ~~used to compute the credential integrity check value in those credentials. If the security version key attribute is changed, the new attribute value should not have been previously used in a credential for the OSD object.~~

The security manager may invalidate credentials for an entire partition by using the SET KEY command (see 6.23) to update the working key version used to compute the credential integrity check value in those credentials.

4.9.6 Request nonces

4.9.6.1 Request nonce format

For some security methods (see 4.9.3), an application client generated request nonce (see table 19) is included in the input data for each integrity check value computation (see 4.9.7) to thwart attempts to capture OSD commands (e.g., FORMAT OSD) and replay them.

Table 19 — Request nonce format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
5				TIMESTAMP				(LSB)
6	(MSB)							
11				RANDOM NUMBER				(LSB)

The TIMESTAMP field contains the number of milliseconds that have elapsed since midnight, ~~January 1,~~ 1 January 1970 UT (see 3.1.47). Timestamp values should be coordinated with the contents of the clock attribute in the Root Information attributes page (see 7.1.2.8) using techniques that are outside the scope of this standard.

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

If the security method being used does not require generation of request nonce values, the nonce TIMESTAMP field should contain zero.

4.9.6.2 Device server validation of request nonces

If the inputs to an integrity check value computation include a request nonce with a non zero timestamp and the nonce has been used in any previous integrity check value computation, 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 NONCE NOT UNIQUE. The command shall be terminated regardless of the success or failure of the previous command in which the duplicate request nonce appeared (e.g., the request nonce appearing in a WRITE command that ultimately fails due to insufficient quota or the request nonce appearing in a CREATE command that ultimately fails because the computed credential integrity check value is wrong shall not be accepted a second time).

If the command is being processed using the CMDRSP security method or the ALLDATA security method (see 4.9.3) and a request nonce with zero in the TIMESTAMP field is received, 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.

Device servers may reduce the amount of resources required to remember every request nonce ever received by comparing the contents of the TIMESTAMP field in each request nonce to the contents of the clock attribute in the Root Information attributes page (see 7.1.2.8).

~~Commands containing~~ If the request nonces with timestamp~~s that are~~ is less than the contents of the clock attribute in the Root Information attributes page minus ~~a delta value may~~ the value in the oldest valid nonce attribute in the

Partition Policy/Security attributes page (see 7.1.2.21), then the command shall be terminated with a CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and with the additional sense code set to NONCE TIMESTAMP OUT OF RANGE. If a command is terminated in this way, the current contents of the clock attribute in the Root Information attributes page shall be returned left-aligned and zero-padded (see 3.7.2) in the COMMAND-SPECIFIC INFORMATION field of the command-specific information sense data descriptor.

~~If the delta value from the contents of the clock attribute in the Root Information attributes page used to enforce minimum timestamp values is a constant, it should be made available to the application client in the oldest valid nonce attribute in the Partition Policy/Security attributes page (see 7.1.2.21). If a range of delta values are used to enforce minimum timestamp values, the smallest value in that range should be made available to the application client in the oldest valid nonce attribute in the Partition Policy/Security attributes page.~~

~~Commands containing~~ If the request nonces with timestamp~~s that are~~ is greater than the contents of the clock attribute in the Root Information attributes page plus ~~a delta value may~~ the value in the newest valid nonce attribute in the Partition Policy/Security attributes page (see 7.1.2.21), 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 NONCE TIMESTAMP OUT OF RANGE. If a command is terminated in this way, the current contents of the clock attribute in the Root Information attributes page shall be returned left-aligned and zero-padded (see 3.7.2) in the COMMAND-SPECIFIC INFORMATION field of the command-specific information sense data descriptor.

~~If the delta value from the contents of the clock attribute in the Root Information attributes page used to enforce maximum timestamp values is a constant, it should be made available to the application client in the newest valid nonce attribute in the Partition Policy/Security attributes page (see 7.1.2.21). If a range of delta values are used to enforce maximum timestamp values, the smallest value in that range should be made available to the application client in the newest valid nonce attribute in the Partition Policy/Security attributes page.~~

~~If the newest valid nonce attribute in the Partition Policy/Security attributes page contains a non-zero value and a command containing a request nonce with a timestamp that is greater than the contents of the clock attribute in the Root Information attributes page plus newest valid nonce attribute value is received, device server may accept the command and treat the nonce as having arrived from far in the future. The minimum number of far in the future nonces accepted shall be made available to the application client in the minimum future requests attribute in the Partition Security attributes page. Optional methods for handling far in the future nonces are described in 4.9.6.3.~~

Successful use of the capability request nonces requires some degree of synchronization between the clocks of the device server and security manager. The protocol for synchronizing the clocks is outside the scope of this standard, however, the protocol should be implemented in a secure manner (i.e., it should not be possible for an adversary to set the clock in the device server backwards to enable the replay of expired credentials).

~~**4.9.6.3 Far-in-the-future nonces**~~

~~**4.9.6.3.1 Introduction**~~

~~Device servers may process commands containing request nonces that exceed the sum of the clock attribute in the Root Information attributes page (see 7.1.2.8) plus newest valid nonce attribute in the Partition Policy/Security attributes page (see 7.1.2.21) and use one or both of the following methods to reduce the amount of resources required to remember every request nonce ever received:~~

~~a) Restricting the use of one or more capabilities based on the contents of the capability AUDIT field as described in 4.9.6.3.2, or~~
~~b) Disabling one or more working keys as described in 4.9.6.3.3.~~

~~4.9.6.3.2 Capability restrictions with far in the future nonces~~

~~To reduce the amount of resources required to remember every far in the future application client nonce ever received, the device server may refuse to accept any additional commands containing a specific combination of capability AUDIT field and capability KEY VERSION field (see 4.x.2.2) values. If the device server takes this action, it should terminate the selected command and all future commands containing the selected combination of capability AUDIT field and capability KEY VERSION field values with a CHECK CONDITION status, a sense key of ILLEGAL REQUEST, and an additional sense code of SECURITY AUDIT VALUE FROZEN.~~

~~The device server may repeat the process described in this subclause as often as necessary to reduce the amount of resources required to implement the duplicate nonce checking requirement stated at the beginning of 4.9.6.2.~~

~~4.9.6.3.3 Working key restrictions with far in the future nonces~~

~~To reduce the amount of resources required to remember every far in the future application client nonce ever received, the device server may refuse to accept any additional commands that use a credential whose capability key has been computed using a specific working key version (see 4.x.2.2) value. If the device server takes this action, it:~~

~~a) Should terminate the selected command and all future commands whose capability key has been computed using a selected working key version value with a CHECK CONDITION status, a sense key of ILLEGAL REQUEST, and an additional sense code of SECURITY WORKING KEY FROZEN; and~~
~~b) Shall set to one the bit in the frozen working key bit mask attribute in the Partition Policy/Security attributes page (see 7.1.2.21) that corresponds to the working key version thus selected.~~

~~The device server may repeat the process described in this subclause as often as necessary to reduce the amount of resources required to implement the duplicate nonce checking requirement stated at the beginning of 4.9.6.2.~~

4.9.7 Integrity check values

An integrity check value is a value produced by a cryptographic function (e.g., HMAC-SHA1) based on a secret key (see 4.9.8) that is able to be computed and verified by the entities knowing the secret key. ~~An integrity~~ Integrity check values are used to verify that:

a) A collection of data fields contain correct values; and
b) The values in those data fields were prepared by the entity that created the integrity check value.

~~If the field specified to contain the integrity check value is has fewer bytes than the output of the cryptographic function, then the cryptographic function output shall be truncated to fit in the field. The output of an HMAC-SHA1 function is 20 bytes. It is common in this standard to specify field sizes of only 12 bytes for integrity check values, meaning that the HMAC-SHA1 function output of 160 bits is truncated into 96 bits.~~

4.9.8 Secret keys

4.9.8.1 Introduction

The hierarchy of secret keys and the mechanisms for updating them are described in:

a) This subclause;
b) The definition of the SET MASTER KEY command, 6.24; and
c) The definition of the ~~defining the~~ SET KEY command, 6.23.

In the OSD security model, the security of transactions depends on a hierarchy of secret keys as shown in table 20, with the highest key in the hierarchy (i.e., the master key) shown at the top of the table and the lowest keys in the hierarchy (i.e., the capability keys) shown at the bottom of the table.

Table 20 — OSD secret key hierarchy

Key Name	Key Shared Using	Key Used To	Key Update Frequency
Keys shared between the security manager and the OSD device server			
Master	SET MASTER KEY command	Update ~~Drive~~ Root key	Change of ~~OBSD (see 3.1.26)~~ logical unit owner
~~Drive~~ Root	SET KEY command	Update Partition key	When Partition key may have been compromised (i.e., very infrequently)
Partition [a]	SET KEY command	Update Working keys	When Working key updates may have been compromised (i.e., infrequently)
Working [b]	SET KEY command	Create Capability keys	When normal key use affords too much chance that the working key might be reverse engineered (i.e., regularly)
Keys shared between the security manager and the application client [c]			
Capability [d]	Credentials and mechanisms not specified in this standard	Secure commands, responses, and data	New with each new Credential

[a] For the purposes of the secret key hierarchy, the root object is treated the same as any other partition OSD object using partition zero.

[b] For each partition, up to sixteen working keys may be active at any time, uniquely identified by the capability KEY VERSION field (see 4.x.2.2).

[c] The device server is capable of computing the capability key (see 4.9.5.4) using the reconstructed credential (see 4.9.5.3).

[d] As a dual purpose number, the capability key is different from other keys in the hierarchy. The capability key is the credential integrity check value.~~, meaning that, even~~ Even though the security manager computes it, the computation is based on values beyond the security manager's control (e.g., the user object to which the credential allows access). ~~The time interval during which the capability key is valid is very short.~~ While changing the working key used to construct the credential integrity check value invalidates the capability key, the credential may expire before that, making the capability key invalid.

~~The OBSD shall save two secret key values for e~~Each master, ~~drive~~ root, and partition~~, and working secret~~ key represents two secret key values as follows:

a) An authentication key that is used to compute the credential integrity check values; and
b) A generation key that is used by future SET KEY commands and SET MASTER KEY commands to compute the updated generation key and new authentication key values.

When an OBSD is manufactured, both the master authentication key and master generation key values shall be provided for each logical unit.~~, but the~~ The ~~two~~ values may be identical.

The secret keys shared between the security manager and OSD device server are very secret information. They should be protected from being discovered by an adversary. They should be stored in a tamper resistant non-volatile manner and may be protected by a tamper resistant software shield. The master key shall be stored in a tamper resistant manner.

The seeds that have been used to create all secret keys other than the master key may be saved in nonvolatile memory for later use in recomputing the secret key values. The OSD logical unit should not store the commands sent to set the master key in a manner that has the potential for being externally accessible.

~~4.9.8.2 Credentials for SET KEY and SET MASTER KEY commands~~

~~Like all other commands, the SET KEY command and SET MASTER KEY command may use a credential and the capability contained in that credential to authorize processing of the command as described in 4.9. These credentials and capabilities are prepared and validated as described in 4.9.4 except as described in this subclause.~~

Editors Note 4 - ROW: The following exceptions are covered in 4.9.5.4.

~~For the SET KEY command (see 6.23), the credential integrity check value is computed using the authentication key of the next higher level in the key hierarchy (see table 21).~~

~~Table 21 — SET KEY integrity check value authentication keys~~

~~Key to update~~	~~Authentication key used to compute the capability key~~
~~Drive Root~~	~~Master~~
~~Partition~~	~~Drive Root~~
~~Working~~	~~Partition~~

~~For the SET MASTER KEY command, the credential integrity check value is computed using the master key authentication key value computed by the previous SET MASTER KEY command.~~

Editors Note 5 - ROW: The following exceptions are covered in 4.x.2.2.

~~For the SET MASTER KEY command:~~

~~a) The credential PARTITION_ID field shall be set to zero; and~~
~~b) The credential OBJECT CREATION CREATED TIME shall be set to either zero or the value in the created time attribute in the Partition Information attributes page (see 7.1.2.9) for partition zero.~~

4.9.8.3 Computing updated generation keys and new authentication keys

The SET KEY command and SET MASTER KEY command 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 command, the generation key from the next higher level in the key hierarchy shall be used (e.g., the ~~drive~~ root key generation key is used to create the first partition keys for a newly created partition), as selected by the KEY TO SET field in the CDB of that command; or
   B) For a SET MASTER KEY command, the previous master key generation key shall be used;
b) The seed value is one of the following:
   A) For a SET KEY command, the contents of the SEED field of the CDB for the command; or
   B) For a SET MASTER command key, the result of the seed exchange step (see 6.20.2);

and

c) The integrity check value algorithm, as specified in the INTEGRITY CHECK VALUE ALGORITHM field in the capability in the CDB for the command.

The updated generation key shall be computed by performing the specified integrity check algorithm with the following inputs:

a) Input key value; and
b) CDB seed value.

The new authentication key shall be computed by performing the specified integrity check algorithm with the following inputs:

a) Input key value; and
b) CDB seed value with the least significant bit changed from zero to one.

4.9.9 OSD security interactions with SPC-3 commands and SAM-3 task management functions

Persistent reservations (see 4.15) are incompatible with an OSD logical unit in which the root object or any partition is using any security method other than NOSEC (see 4.9.3).

Except for the INQUIRY command, the REPORT LUNS command, the REQUEST SENSE command, and the TEST UNIT READY command, all SPC-3 commands are invalid if addressed to an OSD logical unit in which any partition is using any security method other than NOSEC (see table 41 in 6.1). The PERFORM SCSI COMMAND command (see 6.16) allows SPC-3 commands other than persistent reservations commands to be performed under the protection of the current security method.

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 ~~performed~~. The PERFORM TASK MANAGEMENT FUNCTION command (see 6.17) allows SAM-3 task management functions to be processed ~~performed~~ under the protection of the current security method.

5.2 Fields commonly used in OSD commands

OSD commands employ the basic structure shown in table 22 for the service action specific fields (see table 19) so that the same field is in the same location in all OSD CDBs. Fields that are unique to one or two CDBs are not shown in table 22.

Table 22 — OSD service action specific fields

Bit / Byte	7	6	5	4	3	2	1	0
10	OPTIONS BYTE							
11	Reserved		GET/SET CDBFMT		Command specific options			
12	TIMESTAMPS CONTROL							
13 to 15	Reserved							
16	(MSB)							
23	PARTITION_ID							(LSB)
24	(MSB)							
31	USER_OBJECT_ID							(LSB)
32 to 35	Reserved							
36	(MSB)							
43	LENGTH							(LSB)
44	(MSB)							
51	STARTING BYTE ADDRESS							(LSB)
52 to 79	Get and set attributes parameters (see 5.2.1)							
80 to 159	Capability (see 4.x.2.2)							
160 to 199	Security parameters (see 5.2.6)							

5.2.6 Security parameters

The CDB security parameters (see table 23) contain the security information needed for each command.

Table 23 — Security parameters format

Bit / Byte	7	6	5	4	3	2	1	0
				⋮ Other CDB fields				
159								
160	(MSB)			REQUEST INTEGRITY CHECK VALUE				
179								(LSB)
180				REQUEST NONCE				
191								
192				DATA-IN INTEGRITY CHECK VALUE OFFSET				
195								
196				DATA-OUT INTEGRITY CHECK VALUE OFFSET				
199								
				~~Capability (see 4.x.2.2)~~				

…

~~The capability is part of the credential (see 4.9.4) that the application client obtains from the security manager. The device server uses the capabilities along with other information to validate the command as described in 4.9.5.2. The format of a capability is defined in 4.x.2.2.~~

6.19 SET KEY

The SET KEY command (see table 24) causes the OSD device server to update the specified secret key.

Table 24 — SET KEY command

Bit / Byte	7	6	5	4	3	2	1	0
8	(MSB)							
9				SERVICE ACTION (8818h)				(LSB)
10	Reserved							
11	Reserved		GET/SET CDBFMT		Reserved		KEY TO SET	
12	TIMESTAMPS CONTROL							
13				Reserved				
15								
16	(MSB)							
23				PARTITION_ID				(LSB)
24	Reserved				KEY VERSION			
25	(MSB)							
31				KEY IDENTIFIER				(LSB)
32	(MSB)							
51				SEED				(LSB)
52				Get and set attributes parameters (see 5.2.1)				
79								
80				Capability (see 4.x.2.2)				
159								
160				Security parameters (see 5.2.6)				
199								

The GET/SET CDBFMT field specifies the format of the get and set attributes parameters as described in 5.2.1.

The contents of the TIMESTAMPS CONTROL field are defined in 5.2.8.

The KEY TO SET field (see table 25) specifies which key shall be updated, which key identifier shall be stored, and which keys shall be invalid following the SET KEY command.

Table 25 — Key to set code values

Value	Key to update	Key identifier attribute to store	Keys to invalidate
00b	Reserved		
01b	~~Drive~~ Root	The ~~drive~~ root key identifier attribute in the Root Policy/Security attributes page (see 7.1.2.20)	Previous ~~drive~~ root key, and all partition and working keys
10b	Partition	The partition key identifier attribute in the Partition Policy/Security attributes page (see 7.1.2.21)	Previous partition key, and all working keys
11b	Working	The working key identifier attribute in the Partition Policy/Security attributes page selected by the KEY VERSION field in the CDB	None

For every key that is invalidated by a SET KEY command, the associated key identifier attribute shall have its attribute length set to zero.

The contents of the PARTITION_ID field are defined in 5.2.4. If the KEY TO SET field contains 01b and the PARTITION_ID field contains a value other than zero, 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 KEY VERSION field specifies the working key version to be updated. If the KEY TO SET field contains 01b or 10b, the KEY VERSION field shall be ignored.

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 table 25.

The SEED field contains a random number generated from a good source of entropy (e.g., as described in RFC 1750). The updated key values shall be computed as described in 4.9.8.3.

If the least significant bit in the SEED field is set to one, the key values shall not be updated, 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 get and set attributes parameters are defined in 5.2.1. The format of the Data-In Buffer and Data-Out Buffer when attributes are being retrieved or set is described in 4.11.

The security parameters are defined in 5.2.6. The secret key whose authentication key shall be used to compute the capability key for this SET KEY command is specified in ~~4.9.8.2~~ 4.9.5.4.

6.20 SET MASTER KEY

6.20.1 Introduction

The SET MASTER KEY command (see table 26) causes the OSD device server to update the master key secret key.

Table 26 — SET MASTER KEY command

Bit / Byte	7	6	5	4	3	2	1	0
8	(MSB)							
9	SERVICE ACTION (8819h)							(LSB)
10	Reserved							
11	Reserved		GET/SET CDBFMT		Reserved		DH_STEP	
12	TIMESTAMPS CONTROL							
13								
23	Reserved							
24	DH_GROUP							
25	(MSB)							
31	KEY IDENTIFIER							(LSB)
32	(MSB)							
35	PARAMETER LIST LENGTH							(LSB)
36	(MSB)							
39	ALLOCATION LENGTH							(LSB)
40								
51	Reserved							
52								
79	Get and set attributes parameters (see 5.2.1)							
80								
159	Capability (see 4.x.2.2)							
160								
199	Security parameters (see 5.2.6)							

The DH_STEP (Diffie-Hellman step) field (see table 27) specifies which step in the Diffie-Hellman exchange to process.

Table 27 — Diffie-Hellman exchange step values

Code	Description	Reference
00b	Seed exchange	6.20.2
01b	Change master key	6.20.3
10b to 11b	Reserved	

If a SET MASTER KEY command is received with a non-zero value in the DH_STEP field and no SET MASTER KEY command has been received with zero in the DH_STEP field 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.

The usage of other CDB fields is described in the description of each Diffie-Hellman step.

6.20.2 Seed exchange

The GET/SET CDBFMT field specifies the format of the get and set attributes parameters as described in 5.2.1.

The contents of the TIMESTAMPS CONTROL field are defined in 5.2.8.

The DH_GROUP field specifies the coded value selected from the Group Description list of coded values maintained by IANA (see http://www.iana.org/assignments/ipsec-registry) that identifies the DH_group_value to be used for the seed exchange step. If the value in the DH_GROUP field does not appear in one of the DH group attributes in the Root Policy/Security attributes page (see 7.1.2.20) 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 KEY IDENTIFIER field is reserved for the seed exchange step.

The PARAMETER LIST LENGTH field specifies the number of bytes of application client DH_data to be sent to the device server. The application client DH_data is computed as follows:

1) A two-byte or larger random number, x, is generated, observing the requirements in RFC 1750;
2) The application client DH_data is equal to DH_group_value^{x}, where the DH_group value is identified by the code value in the CDB DH_GROUP field.

The ALLOCATION LENGTH field specifies the number of bytes available to receive the device server DH_data (see table 28) sent in response to the SET MASTER KEY command. If the allocation length is not sufficient to contain device sever DH_data, 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.

Table 28 — Device server DH_data format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3				RESPONSE LENGTH (n-3)				(LSB)
4								
n				DEVICE SERVER DH_DATA				

The RESPONSE LENGTH field indicates the number of bytes of device server DH_data that follow.

The DEVICE SERVER DH_DATA field contains the DH_data computed by the device server as follows:

1) A two-byte or larger random number, y, is generated, observing the requirements in RFC 1750;
2) The device server DH_data is equal to DH_group_value^{y}, where the DH_group value is identified by the code value in the CDB DH_GROUP field.

The get and set attributes parameters are defined in 5.2.1. The format of the Data-In Buffer and Data-Out Buffer when attributes are being retrieved or set is described in 4.11.

The security parameters are defined in 5.2.6. The authentication master key (see 4.9.5.4) shall be used to compute the capability key for this SET MASTER KEY command ~~4.9.8.2~~.

After GOOD status has been returned for the SET MASTER KEY command seed exchange step and before the SET MASTER KEY COMMAND change master key step is processed, the next authentication master key and next generation master key shall be computed as described in 4.9.8.3, using a seed value that is the concatenation of the following:

1) DH_group_value$^{xy;}$
2) The contents of the OSD system ID attribute in the Root Information attributes page (see 7.1.2.8);
3) The contents of the product model attribute in the Root Information attributes page;
4) The contents of the serial number attribute in the Root Information attributes page;
5) The contents of the OSD name attribute in the Root Information attributes page; and
6) The contents of the username attribute in the Partition Information attributes page (see 7.1.2.9).

6.20.3 Change master key

The GET/SET CDBFMT field specifies the format of the get and set attributes parameters as described in 5.2.1.

The contents of the TIMESTAMPS CONTROL field are defined in 5.2.8.

The DH_GROUP field is reserved for the change master key step.

The KEY IDENTIFIER field specifies a unique identifier to be associated with the new master key. Successful processing of the SET MASTER KEY command change master key step shall include storing the key identifier value in the master key identifier attribute in the Root Policy/Security attributes page (see 7.1.2.20).

The PARAMETER LIST LENGTH field specifies the number of bytes in the change master key DH_data (see table 29).

Table 29 — Change master key DH_data format

Bit / Byte	7	6	5	4	3	2	1	0
0	APPLICATION CLIENT DH_DATA							
k								
k+1	DEVICE SERVER DH_DATA							
n								

The ALLOCATION LENGTH field is reserved for the change master key step.

The get and set attributes parameters are defined in 5.2.1. The format of the Data-In Buffer and Data-Out Buffer when attributes are being retrieved or set is described in 4.11.

The security parameters are defined in 5.2.6. The next authentication master key computed after the return of GOOD status for the most recent SET MASTER KEY command seed exchange step (see 6.20.2) shall be used to compute the capability key for this SET MASTER KEY command.

Successful processing of a SET MASTER KEY command change master key step shall:

a) Replace the authentication master key with the next authentication master key computed after the return of GOOD status for the most recent SET MASTER KEY command seed exchange step (see 6.20.2);
b) Replace the generation master key with the next generation master key computed after the return of GOOD status for the most recent SET MASTER KEY command seed exchange step;
c) Invalidate all of the following keys (see 4.9.8):
    a) The ~~drive~~ root key;
    b) The partition key for every partition on the OSD logical unit; and
    c) Every working key in every partition on the OSD 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 length set to zero.

7.1.2 OSD attributes pages

7.1.2.1 Attributes pages overview

...

The attributes pages defined by this standard are shown in table 30.

Table 30 — Attributes pages defined by this standard

Page Number	Page Name	Reference	Page Format Defined
0h	User Object Directory	7.1.2.7	No
1h	User Object Information	7.1.2.11	No
2h	User Object Quotas	7.1.2.14	Yes
3h	User Object Timestamps	7.1.2.18	Yes
4h	Collections	7.1.2.19	Yes
5h	User Object Policy/Security	7.1.2.23	Yes
6h to 7Fh	Reserved		
C+0h	Collection Directory	7.1.2.6	No
C+1h	Collection Information	7.1.2.10	No
C+2h	Reserved		
C+3h	Collection Timestamps	7.1.2.17	Yes
C+4h	Reserved		
C+5h	Collection Policy/Security	7.1.2.22	Yes
C+6h to C+7Fh	Reserved		
P+0h	Partition Directory	7.1.2.5	No
P+1h	Partition Information	7.1.2.9	No
P+2h	Partition Quotas	7.1.2.13	Yes
P+3h	Partition Timestamps	7.1.2.16	Yes
P+4h	Reserved		
P+5h	Partition Policy/Security	7.1.2.21	Yes
P+6h to P+7Fh	Reserved		
R+0h	Root Directory	7.1.2.4	No
R+1h	Root Information	7.1.2.8	No
R+2h	Root Quotas	7.1.2.12	Yes
R+3h	Root Timestamps	7.1.2.15	Yes
R+4h	Reserved		
R+5h	Root Policy/Security	7.1.2.20	Yes
R+6h to R+7Fh	Reserved		
F000 0000h to FFFF FFFDh	Reserved		
FFFF FFFEh	Current Command	7.1.2.24	Yes

7.1.2.20 Root Policy/Security attributes page

The Root Security attributes page (R+5h) shall contain the attributes listed in table 31.

Table 31 — Root Security attributes page contents

Attribute Number	Length (bytes)	Attribute	Application Client Settable	OSD Logical Unit Provided
0h	40	Page identification	No	Yes
1h	1	~~Security~~ Default security method	Yes	Yes
2h	6	Oldest valid nonce limit	No	Yes
3h	6	Newest valid nonce limit	No	Yes
~~2h~~ 4h to 5h		Reserved	No	
6h	1	Partition default security method	Yes	Yes
7h	2	Supported security methods	No	Yes
8h		Reserved	No	
9h	6	Adjustable clock	Yes	Yes
Ah to 7FFCh		Reserved	No	
7FFDh	0 or 7	Master key identifier	No	Yes
7FFEh	0 or 7	~~Drive~~ Root key identifier	No	Yes
7FFFh to 7FFF FFFFh		Reserved	No	
8000 0000h to 8000 000Fh	1	Supported integrity check value algorithm	No	Yes
8000 0010h to 8000 001Fh	1	Supported DH group	No	Yes
~~8000 0010h~~ 8000 0020h to FFFF FFFEh		Reserved	No	

The page identification attribute (number 0h) shall have the format described in 7.1.2.2 with the VENDOR IDENTIFICATION field containing the ASCII characters "INCITS" and the ATTRIBUTES PAGE IDENTIFICATION field containing the ASCII characters "T10 Root Security".

The default security method attribute (number 1h) ~~shall identify~~ specifies the security method (see table 36 in 7.1.2.21) used for the processing of the SET KEY command (see 6.23) and SET MASTER KEY command (see 6.24) in the absence of conditions that specify a different security method (see 4.9.m). The value of the default security method attribute shall not be changed by a FORMAT OSD command (see 6.9). The value placed in the default security method attribute when the OBSD (see 3.1.26) is manufactured is vendor specific. If the value of the default security method attribute is changed, the working keys for partition zero should be invalidated using the SET KEY command.

The oldest valid nonce limit attribute (number 2h) specifies the largest value allowed in the oldest valid nonce attribute in any Partition Policy/Security attributes page (see 7.1.2.21).

The newest valid nonce limit attribute (number 3h) specifies the largest value allowed in the newest valid nonce attribute in any Partition Policy/Security attributes page.

The partition default security method attribute (number 6h) specifies the value to be placed in the default security method attribute of each partition, when it is created. The value of the partition default security method attribute shall not be changed by a FORMAT OSD command (see 6.9). The value placed in the partition default security method attribute when the OBSD is manufactured is vendor specific.

The supported security methods attribute (number 7h) indicates which security methods (see 4.9.3) are supported by the OSD logical unit (see table 32).

Table 32 — Supported security methods attribute format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved				ALLDATA	CMDRSP	CAPKEY	NOSEC
1	Reserved							

The NOSEC (NOSEC security method supported) bit is set to zero if the NOSEC security method is not supported. The NOSEC bit is set to one if the NOSEC security method is supported.

The CAPKEY (CAPKEY security method supported) bit is set to zero if the CAPKEY security method is not supported. The CAPKEY bit is set to one if the CAPKEY security method is supported.

The CMDRSP (CMDRSP security method supported) bit is set to zero if the CMDRSP security method is not supported. The CMDRSP bit is set to one if the CMDRSP security method is supported.

The ALLDATA (ALLDATA security method supported) bit is set to zero if the ALLDATA security method is not supported. The ALLDATA bit is set to one if the ALLDATA security method is supported.

The adjustable clock attribute (number 9h) shall contain the current time in use by the OSD device server represented as the count of the number of milliseconds elapsed since midnight, ~~January 1,~~ 1 January 1970 UT (see 3.1.47). The value shall be set to the UT when the OBSD (see 3.1.26) is manufactured and may be modified by the application client after that. The mechanism used to maintain ~~value in~~ the adjustable clock attribute value is outside the scope of the standard. The adjustable clock attribute value should not gain or lose more than one second in any 24-hour interval.

The master key identifier attribute (number 7FFDh) contains the key identifier value from the most recent successful SET MASTER KEY command (see 6.24). If a SET MASTER KEY command has never been processed, the master key identifier attribute length shall be ~~zero~~ seven and the master key identifier attribute value shall be the ASCII characters "1st key".

The ~~drive~~ root key identifier attribute (number 7FFEh) contains the key identifier value from the most recent successful SET KEY command (see 6.23) with the KEY TO SET field set to 01b (i.e., update ~~drive~~ root key). If the ~~drive~~ root key is invalid (i.e., never set or invalidated by a SET MASTER KEY command), the ~~drive~~ root key identifier attribute length shall be zero. Regardless of the root key identifier attribute length, the used capacity attribute in the Partition Information attributes page (see 7.1.2.9) for partition zero (see 3.1.31) shall reflect an attribute length of seven (i.e., it shall not be possible for a SET KEY command to cause the partition zero used capacity attribute value to exceed the capacity quota attribute in the Partition Quotas attributes page (see 7.1.2.13) for partition zero and generate a quote error).

The supported integrity check value algorithm attributes (numbers 8000 0000h to 8000 000Fh) contain coded values (see table 33) identifying the supported algorithms that the OSD device server supports for computing integrity check values. The supported integrity check value algorithm with the lowest valued attribute number (i.e., 8000 0000h) identifies the most preferred integrity check value algorithm and the highest valued attribute number (i.e., 8000 000Fh) identifies the least preferred algorithm. If a supported integrity check value algorithm attribute contains zero, then all supported integrity check value algorithm attributes with higher valued attribute numbers also shall contain zero. The low order four bits of the attribute number are the value that appears in the INTEGRITY CHECK VALUE ALGORITHM field (see 4.x.2.2) in each capability (e.g., attribute number 8000 0007h identifies the integrity check value algorithm used if the INTEGRITY CHECK VALUE ALGORITHM field contains seven).

Table 33 — Supported integrity check value algorithm codes

Value	Algorithm	Reference
00h	No algorithm supported	
01h	HMAC-SHA1	FIPS 180-1 (1995) and FIPS 198 (2002)
02h - DFh	Reserved	
E0h - FFh	Vendor specific	

The supported DH group attributes (numbers 8000 0010h to 8000 001Fh) contain coded values identifying the supported values in the DH_GROUP field of a SET MASTER KEY command (see clause 6.20). The values of the supported DH group attributes are the values associated with a Group Description in the Internet Key Exchange Attributes registry maintained by IANA (see http://www.iana.org/assignments/ipsec-registry). The DH group indicated by each value is as specified by IANA in that registry.

Every DH group identified by a supported DH group attribute shall be a MODP DH group. One of the supported DH group attributes shall contain Dh (i.e., 14) indicating the 2048-bit MODP DH group defined by RFC 3526. The code values 1h (i.e., the 768-bit MODP DH group defined by RFC 2409) and 2h (i.e., the 1024-bit MODP DH group defined by RFC 2409) shall not appear in any supported DH group attribute.

The supported DH group with the lowest valued attribute number (i.e., 8000 0000h) identifies the most preferred DH group and the highest valued attribute number (i.e., 8000 000Fh) identifies the least preferred DH group. If a supported DH group attribute contains zero, then all supported DH group attributes with higher valued attribute numbers also shall contain zero

If a set attributes list (see 5.2.1.3) contains an entry specifying the number of an attribute that table 31 states ~~may not be set~~ is not application client settable, 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 PARAMETER LIST. If the CDB SET ATTRIBUTE NUMBER field (see 5.2.1.2) specifies the number of an attribute that table 31 states ~~may not be set~~ is not application client settable, 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 page format for the Root Security attributes page is shown in table 34.

Table 34 — Root Security attributes page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE NUMBER (R+5h)				
3								(LSB)
4	(MSB)			PAGE LENGTH (~~26h~~ 3Fh)				
7								(LSB)
8				DEFAULT SECURITY METHOD				
9				PARTITION DEFAULT SECURITY METHOD				
10				SUPPORTED SECURITY METHODS				
11								
12	(MSB)			OLDEST VALID NONCE LIMIT				
17								(LSB)
18	(MSB)			NEWEST VALID NONCE LIMIT				
23								(LSB)
24				Reserved			MKI_VALID	~~DR~~KI_VALID
25	(MSB)			MASTER KEY IDENTIFIER				
31								(LSB)
32	(MSB)			~~DRIVE~~ ROOT KEY IDENTIFIER				
38								(LSB)
39				Most preferred SUPPORTED INTEGRITY CHECK VALUE ALGORITHM (attribute number 8000 0000h)				
				⋮				
54				Least preferred SUPPORTED INTEGRITY CHECK VALUE ALGORITHM (attribute number 8000 000Fh)				
55				Most preferred SUPPORTED DH GROUP (attribute number 8000 0010h)				
				⋮				
70				Least preferred SUPPORTED DH GROUP (attribute number 8000 001Fh)				

The PAGE NUMBER field contains the attributes page number of the Root Security attributes page.

The PAGE LENGTH field contains the number of additional bytes in the page format of the Root Security attributes page.

The OLDEST VALID NONCE LIMIT field contains the value of the oldest valid nonce limit attribute.

The NEWEST VALID NONCE LIMIT field contains the value of the newest valid nonce limit attribute.

The DEFAULT SECURITY METHOD field contains the value of the default security method attribute.

The PARTITION DEFAULT SECURITY METHOD field contains the value of the partition default security method attribute.

The SUPPORTED SECURITY METHODS field contains the value of the supported security methods attribute.

The MKI_VALID (master key identifier valid) bit shall be set to zero if the master key identifier attribute length is zero. Otherwise, the MKI_VALID bit shall be set to one.

The ~~DR~~KI_VALID (~~drive~~ root key identifier valid) bit shall be set to zero if the ~~drive~~ root key identifier attribute length is zero. Otherwise, the ~~DR~~KI_VALID bit shall be set to one.

If the MKI_VALID bit is set to one, the MASTER KEY IDENTIFIER field contains the value of the master key identifier attribute. Otherwise, the contents of the MASTER KEY IDENTIFIER field are undefined.

If the DKI_VALID bit is set to one, the ~~DRIVE~~ ROOT KEY IDENTIFIER field contains the value of the ~~drive~~ root key identifier attribute. Otherwise, the contents of the ~~DRIVE~~ ROOT KEY IDENTIFIER field are undefined.

The sixteen ~~16~~ SUPPORTED INTEGRITY CHECK VALUE ALGORITHM fields contain the supported integrity check value attribute values in ascending attribute number order. The SUPPORTED INTEGRITY CHECK VALUE ALGORITHM field with the smallest byte offset in the page identifies the most preferred integrity check value algorithm. The SUPPORTED INTEGRITY CHECK VALUE ALGORITHM field with the largest byte offset in the page identifies the least preferred algorithm.

The sixteen SUPPORTED DH GROUP fields contain the supported DH group attribute values in ascending attribute number order. The SUPPORTED DH GROUP field with the smallest byte offset in the page identifies the most preferred DH group to be used by the SET MASTER KEY command (see clause 6.20). The SUPPORTED DH GROUP field with the largest byte offset in the page identifies the least preferred DH group.

7.1.2.21 Partition Policy/Security attributes page

The Partition Security attributes page (P+5h) shall contain the attributes listed in table 35.

Table 35 — Partition Security attributes page contents

Attribute Number	Length (bytes)	Attribute	Application Client Settable	OSD Logical Unit Provided
0h	40	Page identification	No	Yes
1h	1	~~Security~~ Default security method	Yes	Yes
2h	6	Oldest valid nonce	Yes	Yes
3h	6	Newest valid nonce	Yes	Yes
~~4h~~	~~2~~	~~Minimum future requests~~	~~No~~	~~Yes~~
~~5h~~	~~2~~	~~Frozen working key bit mask~~	~~No~~	~~Yes~~
~~6h~~	~~4~~	~~Security version tag~~	~~Yes~~	~~Yes~~
~~7h~~	~~4~~	~~User object security version tag~~	~~Yes~~	~~Yes~~
~~8h~~ 4h to 7FFEh		Reserved	No	
7FFFh	0 or 7	Partition key identifier	No	Yes
8000h to ~~800Fh~~ 4000 0000h	0 or 7	Working key identifier	No	Yes
4000 0001h	4	Policy access tag	Yes	Yes
4000 0002h	4	User object policy access tag	Yes	Yes
~~8010h~~ 4000 0002h to FFFF FFFEh		Reserved	No	

The page identification attribute (number 0h) shall have the format described in 7.1.2.2 with the VENDOR IDENTIFICATION field containing the ASCII characters "INCITS" and the ATTRIBUTES PAGE IDENTIFICATION field containing the ASCII characters "T10 Partition Security".

The default security method attribute (number 1h) ~~shall identify~~ specifies the security method (see table 36) used for the processing of all commands except the SET KEY command and SET MASTER KEY command in the absence of conditions that specify a different security method (see 4.9.m).

Table 36 — Security method attribute values

Value	Security Method	Reference
00h	NOSEC	4.9.3.2
01h	CAPKEY	4.9.3.3
02h	CMDRSP	4.9.3.4
03h	ALLDATA	4.9.3.5
04h to FFh	Reserved	

A CREATE PARTITION command (see 6.7) shall copy the partition security method attribute from the Root Policy/Security attributes page (see 7.1.2.20) to the security method attribute in new Partition Security attributes page. The value of the security method attribute for partition zero shall not be changed by a FORMAT OSD command (see 6.9). The value placed in the security method attribute for partition zero when the OBSD (see

3.1.26) is manufactured is vendor specific. If the value of the security method attribute is changed, the working keys for affected partition should be invalidated using the SET KEY command (see 6.23).

The oldest valid nonce attribute (number 2h) ~~shall contain~~ indicates the ~~minimum~~ number of milliseconds prior to the value in the clock attribute in the Root Information attributes page (see 7.1.2.8) to which the device server constrains the contents of the TIMESTAMP field in a request nonce (see 4.9.6) received in a command addressed to the partition, a collection in the partition, or a user object in the partition. The processing of request nonces affected by this constraint is described in 4.9.6.2. ~~An oldest valid nonce attribute value of zero indicates that the value is not expressible as a constant.~~

If a set attributes list (see 5.2.1.3) contains a request to set the oldest valid nonce attribute to a value that is larger than the value in the oldest valid nonce limit attribute in the Root Policy/Security attributes page (see 7.1.2.20), 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 PARAMETER LIST. If the CDB SET ATTRIBUTE NUMBER field contains 2h (i.e., the oldest valid nonce attribute) and the set attributes data specified by the SET ATTRIBUTES OFFSET field (see 5.2.1.2) contains a value that is larger than the value in the oldest valid nonce limit attribute in the Root Policy/Security attributes page, 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 newest valid nonce attribute (number 3h) ~~shall contain~~ indicates the ~~minimum~~ number of milliseconds ~~prior to~~ later than the value in the clock attribute in the Root Information attributes page to which the device server constrains the contents of the TIMESTAMP field in a request nonce (see 4.9.6) received in a command addressed to the partition, a collection in the partition, or a user object in the partition. The processing of request nonces affected by this constraint is described in 4.9.6.2 ~~and 4.9.6.3~~. ~~A newest valid nonce attribute value of zero indicates that the value is not expressible as a constant.~~

If a set attributes list (see 5.2.1.3) contains a request to set the newest valid nonce attribute to a value that is larger than the value in the newest valid nonce limit attribute in the Root Policy/Security attributes page, 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 PARAMETER LIST. If the CDB SET ATTRIBUTE NUMBER field contains 3h (i.e., the newest valid nonce attribute) and the set attributes data specified by the SET ATTRIBUTES OFFSET field (see 5.2.1.2) contains a value that is larger than the value in the newest valid nonce limit attribute in the Root Policy/Security attributes page, 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 minimum future requests attribute (number 4h) shall contain the minimum number of commands addressed to the partition, a collection in the partition, or a user object in the partition containing a far in the future request nonce that device server guarantees to process concurrently.~~

~~The frozen working key bit mask attribute (number 5h) indicates which working key versions (see table 37) have been frozen to reduce the amount of resources required to remember every request nonce ever received as described in 4.9.6.3.3.~~

~~**Table 37 — Frozen working key bit mask attribute format**~~

~~**Bit Byte**~~	~~7~~	~~6~~	~~5~~	~~4~~	~~3~~	~~2~~	~~1~~	~~0~~
~~0~~	~~WK07_FZN~~	~~WK06_FZN~~	~~WK05_FZN~~	~~WK04_FZN~~	~~WK03_FZN~~	~~WK02_FZN~~	~~WK01_FZN~~	~~WK00_FZN~~
~~1~~	~~WK0F_FZN~~	~~WK0E_FZN~~	~~WK0D_FZN~~	~~WK0C_FZN~~	~~WK0B_FZN~~	~~WK0A_FZN~~	~~WK09_FZN~~	~~WK08_FZN~~

~~A WK00_FZN (working key 0h frozen) bit set to zero indicates that device server is not rejecting commands that contain credentials with the working key with a key version of zero in order to reduce the resources required to remember every request nonce ever received. A WK00_FZN bit set to one indicates that device server is rejecting~~

~~commands that contain credentials with the working key with a key version of zero in order to reduce the resources required to remember every request nonce ever received as described in 4.9.6.3.3. Once the WK00_FZN bit is set to one, it shall not be set to zero until a new working key with key version zero is established using the SET KEY command (see 6.23).~~

~~The WK01_FZN bit, WK01_FZN bit, WK02_FZN bit, WK03_FZN bit, WK04_FZN bit, WK05_FZN bit, WK06_FZN bit, WK07_FZN bit, WK08_FZN bit, WK09_FZN bit, WK0A_FZN bit, WK0B_FZN bit, WK0C_FZN bit, WK0D_FZN bit, WK0E_FZN bit, and WK0F_FZN have the same bit value definitions as the WK00_FZN bit, except that the definitions apply to the working keys with key versions one to fifteen, respectively.~~

The partition key identifier attribute (number 7FFFh) contains the key identifier value from the most recent successful SET KEY command (see 6.23) with the KEY TO SET field set to 10b (i.e., update partition key). If the partition key is invalid (i.e., never set, invalidated by a SET MASTER KEY command (see 6.24), or invalidated by a SET KEY command), the partition key identifier attribute length shall be zero. Regardless of the partition key identifier attribute length, the used capacity attribute in the Partition Information attributes page (see 7.1.2.9) shall reflect an attribute length of seven (i.e., it shall not be possible for a SET KEY command to cause the partition's used capacity attribute value to exceed the capacity quota attribute in the Partition Quotas attributes page (see 7.1.2.13) and generate a quote error).

The working key identifier attributes (numbers 8000h to 800Fh) contain the key identifier value from the most recent successful SET KEY command with:

a) The KEY TO SET field set to 11b (i.e., update working key); and
b) The KEY VERSION field set to the attribute number minus 8000h (e.g., a version key of three sets attribute 8003h and a version key of eight sets attribute 8008h).

If a working key is invalid (i.e., never set, invalidated by a SET MASTER KEY command, or invalidated by a SET KEY command), the working key identifier attribute length for the associated working key shall be zero. Regardless of the lengths of any of the working key identifier attributes, the used capacity attribute in the Partition Information attributes page shall reflect an attribute length of seven for all sixteen working key identifier attributes (i.e., it shall not be possible for a SET KEY command to cause the partition's used capacity attribute value to exceed the capacity quota attribute in the Partition Quotas attributes page and generate a quote error).

The ~~security version~~ policy access tag attribute (number ~~6h~~ 4000 0001h) specifies the expected non-zero contents of the ~~SECURITY VERSION~~ POLICY ACCESS TAG field in any capability (see 4.x.2.2) that allows access to this partition. The format, use, and attribute setting restrictions for the policy access tag attribute are described in 4.x.3. A CREATE PARTITION command (see 6.7) shall set the ~~security version~~ policy access tag attribute to ~~FFFF FFFFh~~ 7FFF FFFFh.

~~If a set attributes list (see 5.2.1.3) contains an request to set the security version tag attribute to zero, 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 PARAMETER LIST. If the CDB SET ATTRIBUTE NUMBER field contains 6h (i.e., the security version tag attribute) and the set attributes data specified by the SET ATTRIBUTES OFFSET field (see 5.2.1.2) contains zero, 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 user object ~~security version~~ policy access tag attribute (number ~~7h~~ 4000 0002h) specifies the value to be placed in the ~~security version~~ policy access tag attribute of each collection or user object, when it is created. A CREATE PARTITION command (see 6.7) shall set the user object ~~security version~~ policy access tag attribute to ~~FFFF FFFFh~~ 7FFF FFFFh.

~~If a set attributes list (see 5.2.1.3) contains an request to set the user object security version tag attribute to zero, 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 PARAMETER LIST. If the CDB SET ATTRIBUTE~~

~~NUMBER field contains 7h (i.e., the user object security version tag attribute) and the set attributes data specified by the SET ATTRIBUTES OFFSET field (see 5.2.1.2) contains zero, 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 a set attributes list contains an entry specifying the number of an attribute that table 35 states ~~may not be set~~ is not application client settable, 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 PARAMETER LIST. If the CDB SET ATTRIBUTE NUMBER field specifies the number of an attribute that table 35 states ~~may not be set~~ is not application client settable, 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 page format for the Partition Security attributes page is shown in table 38.

Table 38 — Partition Security attributes page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3				PAGE NUMBER (P+5h)				(LSB)
4	(MSB)							
7				PAGE LENGTH (~~8Fh~~ 8Bh)				(LSB)
8								
10				Reserved				
11				DEFAULT SECURITY METHOD				
12	(MSB)							
17				OLDEST VALID NONCE				(LSB)
18	(MSB)							
23				NEWEST VALID NONCE				(LSB)
~~—~~	~~(MSB)~~							
~~—~~				~~MINIMUM FUTURE REQUESTS~~				~~(LSB)~~
~~—~~								
~~—~~				~~FROZEN WORKING KEY BIT MASK~~				
24	(MSB)							
27				~~SECURITY VERSION~~ POLICY ACCESS TAG				(LSB)
28	(MSB)							
31				USER OBJECT ~~SECURITY VERSION~~ POLICY ACCESS TAG				(LSB)
32				Reserved				PKI_VALID
33	WKI07_VLD	WKI06_VLD	WKI05_VLD	WKI04_VLD	WKI03_VLD	WKI02_VLD	WKI01_VLD	WKI00_VLD
34	WKI0F_VLD	WKI0E_VLD	WKI0D_VLD	WKI0C_VLD	WKI0B_VLD	WKI0A_VLD	WKI09_VLD	WKI08_VLD
35	(MSB)							
41				PARTITION KEY IDENTIFIER				(LSB)
42	(MSB)							
60				WORKING KEY IDENTIFIER (for attribute number 8000h)				(LSB)
				⋮				
140	(MSB)							
146				WORKING KEY IDENTIFIER (for attribute number 800Fh)				(LSB)

The PAGE NUMBER field contains the attributes page number of the Partition Security attributes page.

The PAGE LENGTH field contains the number of additional bytes in the page format of the Partition Security attributes page.

The DEFAULT SECURITY METHOD field contains the value of the default security method attribute.

The OLDEST VALID NONCE field contains the value of the oldest valid nonce attribute.

The NEWEST VALID NONCE field contains the value of the newest valid nonce attribute.

~~The MINIMUM FUTURE REQUESTS field contains the value of the maximum future requests attribute.~~

~~The FROZEN WORKING KEY BIT MASK field contains the value of the frozen working key bit mask attribute.~~

The ~~SECURITY VERSION~~ POLICY ACCESS TAG field contains the value of the ~~security version~~ policy access tag attribute.

The USER OBJECT ~~SECURITY VERSION~~ POLICY ACCESS TAG field contains the value of the user object ~~security version~~ policy access tag attribute.

The PKI_VALID (partition key identifier valid) bit shall be set to zero if the partition key identifier attribute length is zero. Otherwise, the PKI_VALID bit shall be set to one.

The WKI00_VLD (working key identifier 0h valid) bit shall be set to zero if the working key identifier attribute number 8000h has a length of zero. Otherwise, the WKI00_VLD bit shall be set to one.

The WKI01_VLD bit, WKI01_VLD bit, WKI02_VLD bit, WKI03_VLD bit, WKI04_VLD bit, WKI05_VLD bit, WKI06_VLD bit, WKI07_VLD bit, WKI08_VLD bit, WKI09_VLD bit, WKI0A_VLD bit, WKI0B_VLD bit, WKI0C_VLD bit, WKI0D_VLD bit, WKI0E_VLD bit, and WKI0F_VLD have the same bit value definitions as the WKI00_VLD bit, except that the definitions apply to the attributes with numbers 8001h to 800Fh, respectively.

The sixteen WORKING KEY IDENTIFIER fields contain the working key identifier attribute values in ascending attribute number order. If a working key identifier valid bit is set to one, the corresponding WORKING KEY IDENTIFIER field contains the value of the working key identifier attribute. Otherwise, the contents of the WORKING KEY IDENTIFIER field are undefined.

7.1.2.22 Collection Policy/Security attributes page

The Collection Security attributes page (C+5h) shall contain the attributes listed in table 39.

Table 39 — Collection Security attributes page contents

Attribute Number	Length (bytes)	Attribute	Application Client Settable	OSD Logical Unit Provided
0h	40	Page identification	No	Yes
1h to ~~5h~~ 4000 0000h		Reserved	No	
~~6h~~	~~4~~	~~Security version tag~~	~~Yes~~	~~Yes~~
4000 0001h	4	Policy access tag	Yes	Yes
4000 0002h to FFFF FFFEh		Reserved	No	

The page identification attribute (number 0h) shall have the format described in 7.1.2.2 with the VENDOR IDENTIFICATION field containing the ASCII characters "INCITS" and the ATTRIBUTES PAGE IDENTIFICATION field containing the ASCII characters "T10 Collection Security".

The ~~security version~~ policy access tag attribute (number ~~6h~~ 4000 0001h) specifies the expected non-zero contents of the ~~SECURITY VERSION~~ POLICY ACCESS TAG field in any capability (see 4.x.2.2) that allows access to this collection. The format, use, and attribute setting restrictions for the policy access tag attribute are described in 4.x.3. A CREATE COLLECTION command (see 6.6) shall copy the user object ~~security version~~ policy access tag attribute from the Partition Policy/Security attributes page (see 7.1.2.21) to the ~~security version~~ policy access tag attribute in new Collection Security attributes page.

~~If a set attributes list (see 5.2.1.3) contains an request to set the security version tag attribute to zero, 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 PARAMETER LIST. If the CDB SET ATTRIBUTE NUMBER field contains 6h (i.e., the security version tag attribute) and the set attributes data specified by the SET ATTRIBUTES OFFSET field (see 5.2.1.2) contains zero, 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 a set attributes list contains an entry specifying the number of an attribute that table 39 states ~~may not be set~~ is not application client settable, 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 PARAMETER LIST. If the CDB SET ATTRIBUTE NUMBER field specifies the number of an attribute that table 39 states ~~may not be set~~ is not application client settable, 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 page format for the Collection Security attributes page is shown in table 40.

Table 40 — Collection Security attributes page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE NUMBER (C+5h)				
3								(LSB)
4	(MSB)			PAGE LENGTH (4h)				
7								(LSB)
8	(MSB)			~~SECURITY VERSION~~ POLICY ACCESS TAG				
11								(LSB)

The PAGE NUMBER field contains the attributes page number of the Collection Security attributes page.

The PAGE LENGTH field contains the number of additional bytes in the page format of the Collection Security attributes page.

The ~~SECURITY VERSION~~ POLICY ACCESS TAG field contains the value of the ~~security version~~ policy access tag attribute.

7.1.2.23 User Object Policy/Security attributes page

The User Object Security attributes page (5h) shall contain the attributes listed in table 41.

Table 41 — User Object Security attributes page contents

Attribute Number	Length (bytes)	Attribute	Application Client Settable	OSD Logical Unit Provided
0h	40	Page identification	No	Yes
1h to ~~5h~~ 4000 0000h		Reserved	No	
~~6h~~	~~4~~	~~Security version tag~~	~~Yes~~	~~Yes~~
4000 0001h	4	Policy access tag	Yes	Yes
4000 0002h to FFFF FFFEh		Reserved	No	

The page identification attribute (number 0h) shall have the format described in 7.1.2.2 with the VENDOR IDENTIFICATION field containing the ASCII characters "INCITS" and the ATTRIBUTES PAGE IDENTIFICATION field containing the ASCII characters "T10 User Object Security".

The ~~security version~~ policy access tag attribute (number ~~6h~~ 4000 0001h) specifies the expected non-zero contents of the ~~SECURITY VERSION~~ POLICY ACCESS TAG field in any capability (see 4.x.2.2) that allows access to this user object ~~collection~~. The format, use, and attribute setting restrictions for the policy access tag attribute are described in 4.x.3. A CREATE command (see 6.4) or CREATE AND WRITE command (see 6.5) shall copy the user object ~~security version~~ policy access tag attribute from the Partition Policy/Security attributes page (see 7.1.2.21) to the ~~security version~~ policy access tag attribute in new User Object Security attributes page.

~~If a set attributes list (see 5.2.1.3) contains an request to set the security version tag attribute to zero, 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 PARAMETER LIST. If the CDB SET ATTRIBUTE NUMBER field contains 6h (i.e., the security version tag attribute) and the set attributes data specified by the SET ATTRIBUTES OFFSET field (see 5.2.1.2) contains zero, 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 a set attributes list contains an entry specifying the number of an attribute that table 41 states ~~may not be set~~ is not application client settable, 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 PARAMETER LIST. If the CDB SET ATTRIBUTE NUMBER field specifies the number of an attribute that table 41 states ~~may not be set~~ is not application client settable, 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 page format for the User Object Security attributes page is shown in table 42.

Table 42 — User Object Security attributes page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE NUMBER (5h)				
3								(LSB)
4	(MSB)			PAGE LENGTH (4h)				
7								(LSB)
8	(MSB)			~~SECURITY VERSION~~ POLICY ACCESS TAG				
11								(LSB)

The PAGE NUMBER field contains the attributes page number of the User Object Security attributes page.

The PAGE LENGTH field contains the number of additional bytes in the page format of the User Object Security attributes page.

The ~~SECURITY VERSION~~ POLICY ACCESS TAG field contains the value of the ~~security version~~ policy access tag attribute.

7.1.2.24 Current Command attributes page

The Current Command attributes page (FFFF FFFEh) shall contain the attributes listed in table 43.

Table 43 — Current Command attributes page contents

Attribute Number	Length (bytes)	Attribute	Application Client Settable	OSD Logical Unit Provided
0h	40	Page identification	No	Yes
1h	~~12~~ 20	Response integrity check value	No	Yes
2h	1	Object Type	No	Yes
3h	8	Partition_ID	No	Yes
4h	8	Collection_Object_ID or User_Object_ID	No	Yes
5h	8	Starting byte address of append	No	Yes
6h to FFFF FFFEh		Reserved	No	

The page identification attribute (number 0h) shall have the format described in 7.1.2.2 with the VENDOR IDENTIFICATION field containing the ASCII characters "INCITS" and the ATTRIBUTES PAGE IDENTIFICATION field containing the ASCII characters "T10 Current Command".

If the NOSEC security method or the CAPKEY security method (see 4.9.3) is used to process the command or if status returned for the command is CHECK CONDITION, the response integrity check value attribute (number 1h) shall contain zero. Otherwise, the response integrity check value attribute shall contain an integrity check value (see 4.9.7) that is computed as described in 4.9.3.4.

NOTE 2 If a command terminates with a CHECK CONDITION status, the response integrity check value is returned in the sense data (see 4.13).

The object type attribute (number 2h) shall identify the type of OSD object on which the current command is operating using the code values shown in table 4 (see 4.x.2.2).

The Partition_ID attribute (number 3h) shall contain the Partition_ID (see 4.6.4) of partition containing the OSD object on which the current command is operating.

If the object type attribute contains COLLECTION (see table 4 in 4.x.2.2), the Collection_Object_ID or User_Object_ID attribute (number 4h) shall contain the Collection_Object_ID (see 4.6.6) of the collection on which the current command is operating. Otherwise, the Collection_Object_ID or User_Object_ID attribute shall contain the User_Object_ID (see 4.6.5) of the user object on which the current command is operating.

If the current command is an APPEND (see 6.2), the starting byte address of append attribute (number 5h) shall contain the starting byte address used for the append command function. If the current command is not an APPEND, the starting byte address of append attribute shall contain zero.

If a set attributes list (see 5.2.1.3) contains an entry specifying the number of an attribute that table 43 states ~~may not be set~~ is not application client settable, 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 PARAMETER LIST. If the CDB SET ATTRIBUTE NUMBER field (see 5.2.1.2) specifies the number of an attribute that table 43 states ~~may not be set~~ is not application client settable, 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 page format for the Current Command attributes page is shown in table 44.

Table 44 — Current Command attributes page format

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			PAGE NUMBER (FFFF FFFEh)				
3								(LSB)
4	(MSB)			PAGE LENGTH (~~28h~~ 30h)				
7								(LSB)
8	(MSB)			RESPONSE INTEGRITY CHECK VALUE				
27								(LSB)
28	OBJECT TYPE							
29				Reserved				
31								
32	(MSB)			PARTITION_ID				
39								(LSB)
40	(MSB)			COLLECTION_OBJECT_ID OR USER_OBJECT_ID				
47								(LSB)
48	(MSB)			STARTING BYTE ADDRESS OF APPEND				
55								(LSB)

The PAGE NUMBER field contains the attributes page number of the Current Command attributes page.

The PAGE LENGTH field contains the number of additional bytes in the page format of the Current Command attributes page.

The RESPONSE INTEGRITY CHECK VALUE field contains the value of the response integrity check value attribute.

The OBJECT TYPE field contains the value of the object type attribute.

The PARTITION_ID field contains the value of the Partition_ID attribute.

The COLLECTION_OBJECT_ID OR USER_OBJECT_ID field contains the value of the Collection_Object_ID or User_Object_ID attribute.

The STARTING BYTE ADDRESS OF APPEND field contains the value of the starting byte address of append attribute.

7.5 Vital product data parameters

7.5.1 Overview

This subclause defines the VPD pages used with OSD type devices.

See SPC-3 for VPD pages used with all device types.

The VPD page codes that are specific to OSD type devices are defined in table 45.

Table 45 — OSD specific VPD page codes

Page code	Description	Reference	Support Requirements
B0h	OSD Information	7.5.2	Optional
B1h	Security Token	7.5.3	Optional
B2h to BFh	Reserved for OSD type devices		

7.5.3 Security Token VPD page

The Security Token VPD page (see table 46) contains a security token for use in the CAPKEY security method (see 4.9.3.3).

Table 46 — Security Token VPD page

Bit / Byte	7	6	5	4	3	2	1	0
0	PERIPHERAL QUALIFIER			PERIPHERAL DEVICE TYPE				
1	PAGE CODE (B1h)							
2	PAGE LENGTH (n-3)							
3								
4	SECURITY TOKEN							
n								

The PERIPHERAL QUALIFIER field and the PERIPHERAL DEVICE TYPE field are defined in SPC-3.

The PAGE LENGTH field specifies the length of the following VPD page data. The page length shall be at least sixteen. If the allocation length is less than the length of the data to be returned, the page length shall not be adjusted to reflect the truncation.

The SECURITY TOKEN field contains a value that is unique to the I_T_L nexus that sent the INQUIRY 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-3) shall cause the security token to change.