ENDL
TEXAS

Date: 29 December 2006
To: T10 Technical Committee
From: Ralph O. Weber
Subject: SNIA requested OSD changes

In June, the SNIA OSD TWG agreed to make the following changes in OSD-2:

A) Adjust Data-In Buffer requirements to permit FCP and SAS support
B) Align CDB fields for better efficiency on 64-bit processors
C) Align attributes list fields for better efficiency on 64-bit processors
D) Clarify when attributes list length may be zero

This proposal contains the detailed OSD-2 text changes that implement the SNIA group's agreements.

In addition, one change not requested by the SNIA OSD TWG is proposed. Because the changes in C) completely redefine the parameter data used by all OSD-2 commands and make it incompatible with the parameter data formats defined in OSD and because the changes in B) reformat the CDBs for several commands, backward compatibility between OSD and OSD-2 can only be maintained by changing the service actions assigned to OSD-2 commands as described in:

E) Change all OSD service actions

The reference OSD-2 revision is r00. Text to be added is show in red. Text to be removed is shown in ~~blue strikeout~~. Commentary text is shown in green.

Change A: Adjust Data-In Buffer requirements to permit FCP and SAS support

The slice-and-dice OSD-2 structure for Data-In Buffers and Data-Out Buffers (see 4.12) is incompatible with the data transfer requirements of FCP and SAS. The incompatibilities have been papered over in OSD and OSD-2 for the Data-Out Buffer, but explicit requirements for Data-In Buffer handling prevent FCP and SAS from being used as OSD-2 (or OSD) transports.

FCP and SAS require all data except the last *n* bytes to be transferred. As currently written, OSD-2 requires up to two untransferred holes in the middle of the Data-In Buffer.

This should be corrected by modifying the last paragraph in 4.12.3 as follows:

~~The~~ If the device server ~~shall not send~~ sends data to the initiator device in the unused Data-In Buffer bytes, then ~~that causes unused bytes in the Data-In Buffer to be overwritten~~ it shall send bytes containing zero.

To improve the clarity of the Data-Out Buffer description, the following changes are recommended in the last paragraph of 4.12.4:

The device server shall ignore the contents of unused bytes in the Data-Out Buffer.

Change B: Align CDB fields for better efficiency on 64-bit processors

There are several 64-bit fields in the CDB that are not aligned at 8 byte boundaries. On a 64- bit Sun SPARC, this is very inconvenient, since attempting to access these fields as 64 bit values will cause an address fault. They have to pull the fields out 32 bits at a time.

The default CDB (see table 41) should be rearranged by moving a the reserved field in bytes 32 – 35 which to immediately before the Get/Set Attributes fields. After this change is made, all 64 bit values fall on 8 byte boundaries.

The following changes should be made in Table 41:

Table 41 — OSD service action specific fields

Byte \ Bit	7	6	5	4	3	2	1	0
10	OPTIONS BYTE (see 5.2.4)							
11	Reserved		GET/SET CDBFMT [a]		Command specific options			
12	TIMESTAMPS CONTROL (see 5.2.8)							
13 – 15	Reserved							
16 – 23	(MSB) PARTITION_ID (see 5.2.5) (LSB)							
24 – 31	(MSB) USER_OBJECT_ID (see 5.2.9) (LSB)							
~~32~~ – ~~35~~	~~Reserved~~							
32~~36~~ – 39~~43~~	(MSB) LENGTH (see 5.2.3) (LSB)							
40~~44~~ – 47~~51~~	(MSB) STARTING BYTE ADDRESS (see 5.2.7) (LSB)							
48 – 51	Reserved							
52 – 79	Get and set attributes parameters [a]							
80 – 159	Capability (see 4.9.2.2)							
160 – 199	Security parameters (see 5.2.6)							

[a] See 5.2.2.

These changes must be applied to each CDB in OSD-2 r00 as follows:

- APPEND — remove reserved bytes 32 – 35 & change reserved bytes 44 – 51 to 40 – 51
- CREATE — remove reserved bytes 32 – 35 & change reserved bytes 38 – 51 to 34 – 51
- CREATE AND WRITE — follow exactly the changes shown in Table 41
- CREATE COLLECTION — no changes needed

- CREATE PARTITION — no changes needed
- FLUSH — no changes needed
- FLUSH COLLECTION — no changes needed
- FLUSH OSD — no changes needed
- FLUSH PARTITION — no changes needed
- FORMAT OSD — change the reserved bytes 13 – 35 to 13 – 31 & change reserved bytes 44 – 51 to 40 – 51
- GET ATTRIBUTES — no changes needed
- LIST — move the LIST IDENTIFIER field from bytes 32 – 35 to bytes 48 – 51
- LIST COLLECTION — move the LIST IDENTIFIER field from bytes 32 – 35 to bytes 48 – 51
- PERFORM SCSI COMMAND — move the reserved bytes from bytes 32 – 35 to bytes 48 – 51
- PERFORM TASK MANAGEMENT FUNCTION — change the reserved bytes 32 – 38 to & add reserved bytes 48 – 51
- READ — follow exactly the changes shown in Table 41
- REMOVE — no changes needed
- REMOVE COLLECTION — no changes needed
- REMOVE PARTITION — no changes needed
- SET ATTRIBUTES — no changes needed
- SET KEY — no changes possible
- SET MASTER KEY — no changes make sense
- WRITE — follow exactly the changes shown in Table 41

These changes must be applied to each CDB in approved proposal 05-328r1 as follows:

- GET MEMBER ATTRIBUTES — no changes needed
- QUERY — move the QUERY LIST LENGTH field from bytes 32 – 35 to bytes 48 – 51
- REMOVE MEMBER OBJECTS — no changes needed
- SET MEMBER ATTRIBUTES — no changes needed

Change C: Align attributes list fields for better efficiency on 64-bit processors

This is essentially the same issue as Change B except that the changes apply to attributes lists (see 7.1.3).

Modify 7.1.3 as follows:

7.1.3 OSD attributes lists

7.1.3.1 Attributes lists overview

An attributes list acts on one or more individual attribute values using attributes page and attribute number values to specify the attribute values to be retrieved or set.

The format of an attributes list is shown in table 126.

Table 126 — Attributes list format

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved				LIST TYPE			
~~1~~	~~Reserved~~							
~~2~~	~~(MSB)~~	~~LIST LENGTH (n-3)~~						
~~3~~								~~(LSB)~~
1	Reserved							
3								
4	(MSB)	LIST LENGTH (n-7)						
7								(LSB)
	Attributes list entries							
8~~4~~	Attributes list entry 0							
	⋮							
	Attributes list entry x							
n								

The LIST TYPE field (see table 127) specifies the format of all attributes list entries in the attributes list.

Table 127 — List type values

List Type	Description	Support Requirement	Reference	Allowed Use		
				Get Attributes		Set Attributes List
				List	Response	
0h	Reserved			No	No	No
1h	Retrieve attributes for this OSD object	Mandatory	7.1.3.2	Yes	No	No
2h - 8h	Reserved			No	No	No
9h	Retrieved/Set attributes for this OSD object	Mandatory	7.1.3.3	No	Yes	Yes
Ah - Eh	Reserved			No	No	No
Fh	Retrieved attributes for a CREATE command (see 6.3) that creates more than one user object	Mandatory	7.1.3.4	No	Yes	No

If list type 1h (see 7.1.3.2) is used to retrieve attributes for this OSD object, the list type of the list containing the retrieved objects shall be:

a) Fh (see 7.1.3.4) for a CREATE command that creates more than one user object; or
b) 9h (see 7.1.3.3) for all other commands and for a CREATE command that creates only one user object.

The LIST LENGTH field indicates the number of bytes of attributes list entries that follow. The LIST LENGTH field may contain zero.

For an attributes list sent from the device server to the application client, a list length of zero indicates that all of the requested attributes have an attribute length of zero.

The application client should set the list length to zero in any attributes list that it sends to the device server. The device server shall use the length of the list specified in the CDB and shall ignore the contents of the LIST LENGTH field.

7.1.3.2 List entry format for retrieving attributes for this OSD object

The attributes list entry format shown in table 128 is used for specifying the attributes to be retrieved by a GET ATTRIBUTES command (see 6.12) or equivalent command function.

Table 128 — List entry format for retrieving attributes for this OSD object

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
				ATTRIBUTES PAGE				
3								(LSB)
4	(MSB)							
				ATTRIBUTE NUMBER				
7								(LSB)

The ATTRIBUTES PAGE field specifies the page number of one attribute to be returned. If the specified attributes page number is not associated with the user object specified by a command, the command shall be terminated with a CHECK CONDITION, with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The ATTRIBUTE NUMBER field specifies:

a) The attribute number within the attributes page specified by the ATTRIBUTES PAGE field of the one attribute value to be returned; or
b) The value FFFF FFFFh to request the return of each attribute having a non-zero attribute length in the attributes page specified by the ATTRIBUTES PAGE field.

If the attribute specified by the ATTRIBUTES PAGE field and ATTRIBUTE NUMBER field has no defined value, an attribute value having a length of FFFF FFFFh shall be returned.

Specifying attributes page and attribute numbers values of FFFF FFFFh causes all defined attributes values in all defined pages associated with the OSD object specified by a command to be returned. Specifying an attribute numbers value of FFFF FFFFh causes all defined attributes values in the specified attributes page to be returned.

If FFFF FFFFh is used as an attributes page number or attribute number value, only those attributes with non-zero lengths shall be returned.

7.1.3.3 List entry format for retrieved attributes and for setting attributes for this OSD object

The attributes list entry format shown in table 129 is used for returning the each attribute value to be retrieved by a GET ATTRIBUTES command (see 6.12) and for specifying each attribute value to be set by a SET ATTRIBUTES command (see 6.21) or equivalent command functions.

Table 129 — List entry format for retrieved attributes and for setting attributes for this OSD object

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
				ATTRIBUTES PAGE				
3								(LSB)
4	(MSB)							
				ATTRIBUTE NUMBER				
7								(LSB)
8	(MSB)							
				ATTRIBUTE LENGTH (n-9)				
9								(LSB)
10	(MSB)							
				ATTRIBUTE VALUE				
n								(LSB)
n+1								
				Pad bytes (for eight-byte alignment)				
m								

The ATTRIBUTES PAGE field specifies the page number of the attribute value.

The ATTRIBUTE NUMBER field specifies the attribute number within the attributes page specified by the ATTRIBUTES PAGE field of the attribute value.

The ATTRIBUTE LENGTH field specifies the length of the attribute value in bytes.

The ATTRIBUTE VALUE field specifies the attribute value.

If the attribute specified by the ATTRIBUTES PAGE field and ATTRIBUTE NUMBER field in a set command function has a defined value, the value shall be replaced by the value specified by the ATTRIBUTE LENGTH field and ATTRIBUTE VALUE field. Otherwise, a new attribute shall be created with the attribute number specified by the ATTRIBUTE NUMBER field in the attributes page specified by the ATTRIBUTES PAGE field.

If the ATTRIBUTES PAGE or ATTRIBUTE NUMBER field contains FFFF FFFFh for a set command function, the command shall be terminated with a CHECK CONDITION, with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The list entry length shall be a multiple of eight bytes. Depending on the attribute length, zero to seven bytes containing zeros shall be added at the end of the list entry to meet the length requirement.

7.1.3.4 List entry format for attributes retrieved by CREATE command that creates multiple user objects

The attributes list entry format shown in table 130 is used for returning each attribute value for each user object requested by a CREATE command (see 6.3) that creates more than one user object.

Table 130 — List entry format for attributes retrieved by a CREATE command creating multiple user objects

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	USER_OBJECT_ID						
7								(LSB)
8	(MSB)	ATTRIBUTES PAGE						
11								(LSB)
12	(MSB)	ATTRIBUTE NUMBER						
15								(LSB)
16	(MSB)	ATTRIBUTE LENGTH (n-17)						
17								(LSB)
18	(MSB)	ATTRIBUTE VALUE						
n								(LSB)
n+1	Pad bytes (for eight-byte alignment)							
m								

The USER_OBJECT_ID field indicates the User_Object_ID of the user object (see 4.6.5) associated with the attribute value.

The ATTRIBUTES PAGE field indicates the page number of the attribute value.

The ATTRIBUTE NUMBER field indicates the attribute number within the attributes page specified by the USER_OBJECT_ID field and ATTRIBUTES PAGE field of the attribute value.

The ATTRIBUTE LENGTH field indicates the length of the attribute value in bytes.

The ATTRIBUTE VALUE field indicates the attribute value.

The list entry format described in this subclause shall not be returned for any command other than a CREATE command that creates more than one user object.

If the list entry format described in this subclause appears in a SET ATTRIBUTES command (see 6.21) or equivalent command function, the command shall be terminated with a CHECK CONDITION, with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The list entry length shall be a multiple of eight bytes. Depending on the attribute length, zero to seven bytes containing zeros shall be added at the end of the list entry to meet the length requirement.

Change D: Clarify when attributes list length may be zero

As is the customary practice in SCSI, device servers are not allowed to truncate list lengths to reflect the effects of parameter truncation due to allocation length being exceeded. However, application clients are usually allowed to send lists of zero length.

The second sentence in the LIST LENGTH field definition (see 7.1.3.1) does not reflect this nuance.

Modify the LIST LENGTH field definition as follows:

The LIST LENGTH field indicates the number of bytes of attributes list entries that follow. For attributes lists sent from the application client to the device server, the ~~The~~ LIST LENGTH field may contain zero.

Change E: Change all OSD service actions

Because the attributes list format changes (see Change C) affect the parameter data for all OSD-2 commands, the only way to maintain backwards compatibility between OSD and OSD-2 is to use new service action code values for OSD-2. (Note that the CDB format changes (see Change B) also are a backwards compatibility issue, but the parameter data format changes are more pervasive.)

ModifyTable B.1 as follows and change all service action codes in the normative body to match these changes, also change the service action codes in 05-328r1 to match those shown here:

Table B.1 — Numerical order OSD service action codes

Service Action	Command
8801h	~~FORMAT OSD~~ Obsolete
8802h	~~CREATE~~ Obsolete
8803h	~~LIST~~ Obsolete
8804h	Reserved
8805h	~~READ~~ Obsolete
8806h	~~WRITE~~ Obsolete
8807h	~~APPEND~~ Obsolete
8808h	~~FLUSH~~ Obsolete
8809h	Reserved
880Ah	~~REMOVE~~ Obsolete
880Bh	~~CREATE PARTITION~~ Obsolete
880Ch	~~REMOVE PARTITION~~ Obsolete
880Dh	Reserved
880Eh	~~GET ATTRIBUTES~~ Obsolete
880Fh	~~SET ATTRIBUTES~~ Obsolete
8810h to 8811h	Reserved
8812h	~~CREATE AND WRITE~~ Obsolete
8813h to 8814h	Reserved
8815h	~~CREATE COLLECTION~~ Obsolete
8816h	~~REMOVE COLLECTION~~ Obsolete
8817h	~~LIST COLLECTION~~ Obsolete
8818h	~~SET KEY~~ Obsolete
8819h	~~SET MASTER KEY~~ Obsolete
881Ah	~~FLUSH COLLECTION~~ Obsolete

Table B.1 — Numerical order OSD service action codes

Service Action	Command
881Bh	~~FLUSH PARTITION~~ Obsolete
881Ch	~~FLUSH OSD~~ Obsolete
881Dh to 8880h	Reserved
8881h	FORMAT OSD
8882h	CREATE
8883h	LIST
8884h	Reserved
8885h	READ
8886h	WRITE
8887h	APPEND
8888h	FLUSH
8889h	Reserved
888Ah	REMOVE
888Bh	CREATE PARTITION
888Ch	REMOVE PARTITION
888Dh	Reserved
888Eh	GET ATTRIBUTES
888Fh	SET ATTRIBUTES
8890h to 8891h	Reserved
8892h	CREATE AND WRITE
8893h to 8894h	Reserved
8895h	CREATE COLLECTION
8896h	REMOVE COLLECTION
8897h	LIST COLLECTION
8898h	SET KEY
8899h	SET MASTER KEY
889Ah	FLUSH COLLECTION
889Bh	FLUSH PARTITION
889Ch	FLUSH OSD
889Dh to 889Fh	Reserved
88A0h	QUERY
88A1h	REMOVE MEMBER OBJECTS
88A2h	GET MEMBER ATTRIBUTES
88A3h	SET MEMBER ATTRIBUTES
~~881Dh~~ 88A4h to 8F7B~~D~~h	Reserved
8F7Ch	PERFORM SCSI COMMAND
8F7Dh	PERFORM TASK MANAGEMENT FUNCTION
8F7Eh	~~PERFORM SCSI COMMAND~~ Obsolete
8F7Fh	~~PERFORM TASK MANAGEMENT FUNCTION~~ Obsolete
8F80h to 8FFFh	Vendor specific