TO: X3 Secretariat
ATTN: Lynn Barra
1250 Eye Street NW, Suite 200
Washington, DC 20005

FROM: Thomas Wicklund
Fujitsu Computer Products of America
1506 Harvard Street
Longmont, CO 80503

(303)-682-6549
FAX (303)-682-6401
email wicklund@intellistor.com

Public review comments of X3.276:199x, SCSI-3 Controller Commands (SCC)

I am working from document X3T10/1047D revision 4.

As a preliminary note, I use the RAID levels as defined in the Berkeley papers. I realize SCC doesn't explicitly define RAID levels but it was easiest to describe common configurations.

General comments:

Interfaces used for devices: There is an assumption throughout the standard that the back-end interface of a SCSI-3 storage array is also SCSI. If this is intended as a requirement it should be stated up front. If not a statement should be made about non-SCSI back ends.

Types of devices: There is also an assumption that a SCSI-3 storage array is an array of disk drives (or more generally SCSI block devices). This is not stated anyplace I can find but is implied by the Normative Reference in clause 2 to SBC and the extensive use of logical block addresses. If devices must be block devices, this should be stated, alternately a statement added that other device types (e.g. sequential) may be used and the user can figure out a reasonable way to map the device into the array.

The problem I see here would be an array device that also has the ability to connect a tape backup or logging device. There is no clear way to configure a non-block device into a storage array.

Data mapping complexity: The SCC standard provides a very flexible, complex two level data mapping scheme. One level maps physical blocks to user or check data (converting a set of p_extents to a set of ps_extents within a redundancy group), while the second maps data into volume sets (ps_extents to volume sets).

Most of the parameters which specify the data mapping are specified independently, allowing a different value for each unit being mapped together. However, in practice most RAID devices are going to define relationships between these parameters. If arbitrary parameters are allowed (and not all combinations of parameters work), the LBA mapping

algorithm becomes extremely complex.

In addition, I don't see any way for a host to determine what parameters a SACL supports, or what relationships are supported.

I think the complexity of the SCC standard is going to make it unusable. Creating a generalized array configuration utility for a host will either involve a user manually entering the restrictions the SACL requires or a lengthy trial and error process.

As a simple example for a single parameter, take the GRANULARITY OF UNITS field which is used both when defining redundancy groups and volume sets. This field has values for bit, byte, 2-byte, 4-byte, and logical block granularity (in addition to vendor specific). I doubt many real world array devices support all of these values (all I'm aware of support only one or two values). The implementer of a SACL is left with a choice -- either accept any value and use what the hardware supports (e.g. treat bit, byte, 2-byte, and 4-byte as the single hardware sub-logical block mapping) or require that the host keep trying values until one happens to work (which becomes much more difficult if a RAID-3 device happens to use an 8-byte word, for example, and to be honest should report Vendor Specific).

In redundancy groups, common RAID devices use the same NUMBER OF UNITS OF CHECK DATA and NUMBER OF UNITS OF USER DATA fields for all p_extents in a redundancy group. In volume sets, the USER DATA STRIPE DEPTH (specified individually per ps_extent) and NUMBER OF UNITS OF USER DATA (specified individually in the underlying p_extents) normally need a defined relationship between them or the mapping from volume set LBA to p_extent LBA becomes convoluted. Yet each parameter is individually specified.

I think this complexity is going to leave SCC unusable. I know it was much harder to figure out than most standards. I think a simpler method of array configuration needs to be provided as an option or the flexibility of SCC should be reduced so that the majority of products don't need to deal with the complexity of checking all of redundant SCC configuration parameters.

Related to the complexity issue, there is no way for a host to determine the capabilities of an arbitrary SACL other than trial and error. Given the number of parameters which can be specified, it isn't practical to produce a general array device driver, leaving one little better than the current vendor specific solutions.

There is also no way to determine possible component device attachments. It appears a host must try to attach a component device to a logical unit and see what happens.

Specific comments by page and clause:

Page 3, Clause 3.1.14, initiator: Does this definition include SACL devices? The diagram on page 10 says an SACL can be an initiator, but it doesn't really contain an application client. The diagram on page 11 specifies the HBA as the initiator, but is the application client limited to the HBA or does the SAM definition encompass the total host system (HBA and above) as the application client?

Page 4, Clause 3.1.39, target: A target is not restricted to a SCSI-3 storage array device in this standard. In fact, on page 11 the diagram labels one box with "drive (target)". This definition should be expanded to include drives as targets.

Page 19, clause 5.2.1.4, Peripheral device address method: The second paragraph on page 19 starting "The TARGET/LUN field indicates ..." is not clear:

- If the BUS NUMBER is 0, the TARGET/LUN field refers to a LUN within the current level of the storage array. What is a LUN? an SACL defines volume sets, which are already addressable. For SIP only, a mapping is defined via a mode page to 5 bit LUN numbers. I can't find a definition of a LUN at the SACL level which makes sense here.

Page 22, clause 5.2.2.10, P_extent: Is this clause correct? The second paragraph states that "a single p_extent that contains no check data may be configured into one or more redundancy groups." Does this mean that an assigned p_extent which contains no check data can be configured into another redundancy group but if it contains no check data it can't?

I think the paragraph should specify that an "unassigned p_extent may be configured into one or more redundancy groups or one or more spares". As written, p_extents which contain check data may only be configured into spares (since they can't be configured into redundancy groups), which sounds wrong.

Should this clause contain a statement about how an application client determines unassigned p_extents (analogous to the next clause on ps_extents)?

Page 24, clause 5.2.2.12, Redundancy Group; also page 80-82, clause 6.4.1.2, Create/Modify Redundancy Group service action:

The NUMBER OF UNITS OF CHECK DATA and NUMBER OF UNITS OF USER DATA fields are allowed to differ in each p_extent within a redundancy group. Are there real examples of RAID configurations which require this? The only example (in Annex C) uses the same value for each p_extent, and in fact the example specifies the values once rather than once per p_extent.

In the subclauses of this clause all of the standard data mappings appear to require that NUMBER OF UNITS OF USER DATA be equal for all p_extents, which implies that NUMBER OF UNITS OF CHECK DATA be equal for all p_extents. Vendor specific redundancy mapping may require a different set of parameters (which SCC doesn't provide a means of specifying).

Similarly there are specific requirements on the START CHECK DATA INTERLEAVE UNIT for each device. This field is either 0 for all p_extents or must define a RAID-5 rotating parity scheme.

Looking further at each mapping defined by SCC, these parameters aren't consistent in the mappings where they don't matter. For the no redundancy method, why is NUMBER OF UNITS OF USER DATA set to 0? It seems like it should be labeled as "set to desired value", whatever that might be.

There is some justification to copy redundancy leaving NUMBER OF UNITS OF CHECK DATA at 0, since each p_extent replicates the user data.

For XOR redundancy the NUMBER OF UNITS OF USER DATA must be equal for all p_extents. Yet for a traditional RAID-3 device I would think that the check data p_extent would use NUMBER OF UNITS OF USER DATA equal to zero and NUMBER OF UNITS OF CHECK DATA non-zero. Otherwise the host cannot specify which p_extent contains check data, which seems odd since it specifies exactly where check data belongs in a RAID-5 type mapping.

If these parameters must remain as they are, an example of how different values between p_extents are useful would help.

Page 25, Figure 12: This flow chart only works for a RAID-5 type data mapping. If the parameters required for "no redundancy" or "copy redundancy" mapping are plugged in the result is incorrect since the first block of each p_extent becomes flagged as "check data" and the "units of check data counter" decrements below zero, which doesn't make sense in the context of the flow chart. A similar problem occurs in a RAID-3 type or XOR mapping.

Page 26, clause 5.2.2.12.3, XOR or P+Q redundancy...: The standard should define XOR and P+Q mapping. Without a definition it appears a product may use any check data mapping method it likes and call it either XOR or P+Q, so long as the standard parameters are used to define the mapping.

The RAID Advisory Board has defined these terms, but no RAB documents are listed in the references to the SCC standard.

The RAB definition of P+Q mapping is two check blocks for N data blocks allowing operation after 2 failures. A P+Q mapping using rotating parity with an odd number of p_extents cannot be specified using the NUMBER OF UNITS OF CHECK DATA and NUMBER OF UNITS OF USER DATA parameters.

Page 28, Figure 14: An additional figure showing multiple volume sets associated with one redundancy group will help illustrate that there is no relationship between redundancy groups and volume sets since the two are assigned independently.

Page 30 and 31, Figures 15 and 16: This algorithm should include the effect of setting the INCDEC field to 1 or state that it's a limited example rather than a "general implementation".

Page 31, bottom of page: The statement that "Figure 17 shows the most general implementation of the three parameters used to map the user data" is not true. Figure 17 uses a constant "user data stripe depth" while the Create/Modify Volume Set service action allows it to be different for each ps_extent.

An example showing the utility of different "user data stripe depth" values in each ps_extent should be added or this value should be made constant for all ps_extents in a volume set.

Page 33, clause 5.2.3.4, Verify Action: This operation is more properly handled by the existing VERIFY command. If a volume set or redundancy group uses a peripheral device type which doesn't define the VERIFY command, then either VERIFY should be added to that peripheral device type or the array verify operation is not appropriate.

Page 35, Table 8: First, this table is mis-labeled as applying to disk storage devices. Second, it seems like the RECEIVE DIAGNOSTIC RESULTS command should be allowed, along with READ BUFFER and possibly LOG SELECT / LOG SENSE.

Page 39, clause 6.1.1.1: Under NUMBER OF BYTES PER LBA_P, I don't think a value of 0 for a stream device makes sense since most current stream devices use fixed block sizes. This isn't required since stream devices are already identified by the peripheral device type of the assigned/unassigned p_extent descriptor.

Page 55, table 34, Logical Unit States Descriptor: Why is the REPLACE bit replicated on each state line? Since this descriptor describes one logical unit, it seems like it's either replaceable or not replaceable. Move the REPLACE bit to byte 1 or 4, or 5 of the descriptor and allow the state to be an 8 bit value. Otherwise explain how a logical unit can have different REPLACE bit values for different states returned.

Page 56, clause 6.1.1.7, Report States: The last sentence of this clause states that a target is operational if the READYING, ABNORMAL, and NONAFAIL bits are 0. If a p_extent fails and is rebuilt to a spare, the ABNORMAL bit should be set to 1 (since an addressable device has a state other than available. This means a target is not operational if any component (spare or not) has failed. If this is intended, a term other than "operational" should be chosen since most users will consider an array which executes reads and writes to be "operational" even if data needs to be rebuilt or a spare needs to be replaced.

At the very least, define the term "operational".

Page 57, Table 36: Move state Protection disabled into numeric order with the rest of the codes or fix the table to be alphabetical. It appears an attempt was made to make tables 36-40 alphabetical but tables 36 and 37 are not.

Capitalization should also be made consistent in the state names.

Page 57, Table 36, state "Exposed": Change the description from "a failure causes" to "a failure may cause". A volume set can contain multiple redundancy groups and only one may be exposed, while the others are not. Whether data is lost depends on which component fails.

Page 57, Table 36, state "Readying", note 13: According to this note, if the underlying drives of a volume set are not spinning (waiting for

a START UNIT) or otherwise not ready the volume set state is "Readying", which seems to be inaccurate. Either add a volume set state for "Not Ready" (or perhaps "Present") or be sure the "Readying" description clearly states that the volume set may not become available without outside action.

Page 57, Table 36, states "Recalculate" and "Verify in progress": Change these description to include the case where one or more underlying redundancy groups is being recalculated or verified but not the full volume set (analogous to the Rebuild state).

Page 58, table 37, state "Invalidated Protected Space": Why not use "Data lost" like volume sets? Alternately, change the volume set state to be "Invalidated Protected Space".

Page 58, table 37: Shouldn't there be a "Spare in use" state as in volume sets? Without this state can a volume set easily detect the "Spare in use" state by querying its redundancy groups?

Page 60, end of clause 6.1.1.7: Move the last paragraph about the replace bit to before all of the state tables. At this point the reader has forgotten about the structure being defined.

Page 62, top of page: Reference the tables defining the different DEVICE TYPE values.

Page 62, clause 6.2.1.2, "ATTACH TO COMPONENT DEVICE service action": The first sentence in this clause refers to a logical unit in the singular, but the service action can attach multiple logical units to a component device.

Page 80, clause 6.4.1.1, ALLRG bit. Also page 98, clause 6.6.1.1, ALLVLU: The description of the ALLRG (ALLVLU) bit is wrong. Zero or one define whether the DISCHK bit applies to all redundancy groups or the addressed redundancy group, the current definition conflicts with DISCHK.

In addition, the description of DISCHK for both these clauses doesn't define what happens if checking is disabled for a redundancy group then enabled for the volume set containing the redundancy group. Each description states that after being disabled check data isn't re-enabled until done so for the like structure (e.g. disable check data on a redundancy group, must enable for a redundancy group). As stated the interaction is ambiguous.

Page 83, clause 6.4.1.2, Protected Space Pattern: Shouldn't this be a variable length field similar to the FORMAT command for disk drives? I assume there was a good reason why a byte fill field wasn't used for disks and it would seem to apply here also.

Page 84, clause 6.4.1.4, REBUILD P_EXTENT service action: The first sentence should include the possibility of rebuilding several redundancy groups (both because of overlapping redundancy groups and because a p_extent may consist of several redundancy groups).

Page 85, Table 73, Rebuild types:

A. Why do codes 00b and 10b specify that assigned space be rebuilt while for code 01b the full p_extent is apparently rebuilt whether assigned or not? Shouldn't code 01b also specify assigned space, or perhaps text outside the table should say that this service action "rebuilds the assigned space associated with the p_extent"?

B. The second paragraph for each rebuild type (If the rebuild operation fails...) should be removed from the table and made an independent paragraph since it's identical for all rebuild codes.

C. I think it would be better to require that the list of redundancy groups be empty for codes 00b and 01b. This is better than the confusion which could result from ignored parameters.

Page 86, clause 6.4.1.5, REBUILD PERIPHERAL DEVICE service action: The first sentence of this clause should include the fact that several redundancy groups might be rebuilt.

Page 88, Table 76: My earlier comments about table 73 also apply here.

Page 98, clause 6.6.1.2, CONTROL WRITE OPERATIONS service action: SCSI-2 already went through the question of whether to allow the host to set write protect for a disk drive or not. This service action is more properly added to the SBC or similar standard as a write protect capability.

This clause should also define whether rebuild and regenerate operations are considered writes and whether a Modify or Delete action is considered a write.

Page 100, clause 6.6.1.3, CREATE/MODIFY VOLUME SET service action: The definitions at the bottom of page 100 (and also earlier in the model) use the term "stripe". It ought to be defined in the glossary.

At the bottom of page 100 it states that the PS_EXTENT STRIPE LENGTH must be an exact multiple of the number of ps_extents. I think this is backwards, it should be that "the number of ps_extents is an exact multiple of the PS_EXTENT STRIPE LENGTH".

Page 101, top of page: The term "PS_EXTENT LENGTH field" is used. I think "PS_EXTENT" should be lower case here (or it should be upper case elsewhere in the text). Looking back at table 65, I can't find a "LENGTH" field associated with a ps_extent descriptor. The "PS_EXTENT LENGTH field" should be renamed to use the correct field name.

Page 101, PS_EXTENT INTERLEAVE DEPTH: I can't figure out what to do with this field from the text. I can sort of figure it out from figure 17, but it isn't clear what the units are. Figure 20 makes the use of this field apparent but the standard shouldn't depend on an informative annex.

Page 101, Table 90: This table is identical to table 84, should they be combined?

Page 101, PS_EXTENT DESCRIPTOR, INCDEC field: Is this really necessary? I can't see any good reason why this feature is needed. In fact, until disk drives have INCDEC fields setting this field to 1 would appear to be a guaranteed performance killer by trying to read and write the drive backwards.

Also in this paragraph, it states "When INCDEC is one logical blocks after the START LBA_PS field in the PS_EXTENT DESCRIPTOR shall be assigned in descending order". This seems to say assign the first block to START LBA_PS, then assign the rest in descending order.

Page 103, clause 6.6.1.5, RECALCULATE VOLUME SET CHECK DATA service action: Why doesn't this service action have the three options specified in the VERIFY RANGE of the next clause? Minor inconsistencies like this make using a standard interface very frustrating since one is constantly annoyed that an option isn't available for a particular operation and there's no obvious reason for it.

Page 105, clause 6.6.1.6, VERIFY VOLUME SET CHECK DATA service action: Why not redefine VERIFY RANGE so that bit 1 of byte 10 is an ALLVLU bit? Then use bit 3 to specify whether the selected LBA range should be used or not.

Page 106, first paragraph: According to this paragraph, if a volume set consists of 4 redundancy groups, and one redundancy type "no redundancy", the other redundancy groups must be verified using a VERIFY REDUNDANCY GROUP service action rather than the volume set version. Why not just state that redundancy groups with no redundancy are ignored, especially if the LBA range being verified doesn't include the "no redundancy" portion of the volume set.

Page 106, CONTVER bit: What's the interaction between the CONTVER bit in a volume set and the CONTVER bit of a redundancy group? If independent, it appears that a SACL must keep a CONTVER flag for each ps_extent (since a redundancy group may have one setting while each volume set using it may have another) and a bit for each redundancy group the ps_extent is part of.

Page 123, Annex A: The IDENTIFY message defined here uses bit 5 for the VOLSEL bit. The SIP standard specifies that bit 5 is reserved. Either SCC should not use bit 5 (reduce the LUN size to 4 bits) or SIP should be changed to remove the reserved bit from IDENTIFY.

Page 132, Figure 20: What is a "strip depth" (I assume you mean "stripe depth")?

This figure should state that an LBA_PS is 512 bytes (I think) and that an LBA_V is 512 bytes (though there's no way to specify this within SCC).