July 5, 2001

To: John Lohmeyer, T10 Chairman
Ralph Weber, SPC-2 Technical Editor
From: Roger Cummings, VERITAS Software
Subject: Comments on the Response to the VERITAS Public Review Comment on dpANS SCSI Primary Commands -2

John,

VERITAS would like to thank Ralph Weber for his consideration of our Public Review comment against FCP-2 (01-175r0), and his response (contained in 01-194r2).

VERITAS has comments on the response, which are listed below. We are requesting an agenda item of 30 minutes in length at the next meeting of the SCSI Commands Architecture and Protocols Working Group to discuss the response and these comments in detail. Subsequent to the presentation, VERITAS will be pleased to accept the recommendation of the CAP WG, and the decision of the subsequent Technical Committee T10 Plenary, on this subject.

Regards,

Roger Cummings
VERITAS Software
roger.cummings@veritas.com
407.531.7257

VERITAS COMMENTS ON RESPONSES TO OUR PUBLIC REVIEW COMMENT on SPC-2 CONTAINED in 01-194r2

1) The first paragraph of response to the VERITAS Public Review comment states:

The changes requested substantially alter the behavior intended by T10 when defining the Persistent Reservations feature. The intent of T10 is that releasing a reservation should be part of a controlled software shutdown process during which cooperating members of a cluster should use Persistent Reservation commands to gracefully transfer reservations ownership to remaining cluster members. The behavior described in SPC-2 is consistent with that intent.

This wording is at odds with the overview of the Persistent Reservation management method in SPC-2. The first and second paragraphs of subclause 5.5.3.1 state:

The persistent reservations management method is the mechanism specified by this standard for use by multiple initiators that require operations to be protected across initiator failures, which usually involve hard resets. ...... Persistent reservations for failing initiators may be preempted by another initiator as part of the recovery process.

The text quoted above is an accurate reflection of the behavior of persistent reservations. Underlying the behavior is an assumption of a failure mechanism where an Initiator is unable to perform cleanup activities such as deregistration. Nowhere in the text in SPC-2 is there a reference to a controlled software shutdown process. In such a process, it would surely be expected that an Initiator seek to release the system resources that have been dedicated to it, in the interests of efficiency and system scalability. One of such resources is the registration held by a device server. While the actual resources consumed by the registration may be trivial, because there is no indication of the maximum number of reservations that a device can support the removal of such a registration is good policy. However using the existing definition, the impact of the removal on the remaining initiators is significant and, more than that, offers an advantage to an initiator that did not participate in the original reservation in trying to establish a new reservation, with the resulting opportunity for data corruption.

2) The second and third paragraphs of response to the VERITAS Public Review comment state:

Furthermore, changing the definition in this way at this late date will materially harm those who worked with T10 during the process of defining Persistent Reservations and as such would be a disservice to the industry.

If the new behavior proposed is desired, it should be proposed for SPC-3 as a new Persistent Reservation Type (e.g., Write Exclusive - All Registrants).

We understand the concern with changing an existing "code point" at this late date, but chose to word our Public Review comment that was for the following reasons:

a) We did not feel it was appropriate to attempt to introduce new functionality at this stage in the process, rather we only sought to correct what we saw as an issue with an existing definition;

b) We believed that the change being requested was relatively minor and self-contained;

c) None of our partners had told us that the current definition of releasing a reservation upon deregistration of the initiator that made the reservation was a “must have” function.

However since our Public Review comment was posted, a number of people have expressed some concern to us regarding the change in definition at this stage in the process. While sympathizing with those concerns, we are somewhat doubtful about establishing a precedent of those concerns being the ONLY grounds for the rejection of a Public Review comment.

SUMMARY

The VERITAS position with regards to the Public Review comment and its response can be summarized as follows:

a) We regret that the current definition of Registrants Only Persistent Reservations makes them unsuitable for use in our clustering applications, due to the possibility for data corruption when the initiator that made the reservation deregisters. This is unfortunate as the other features of Persistent Reservations offer us a significant improvement upon our current approach using Reserves and Releases.

b) We have not yet heard that the current functionality is a “must have”, only that the redefinition of an existing code point at this stage in the development of a dpANS is a cause for concern. We sympathize, but doubt that this being the only grounds for rejection of a Public Review comment is a good precedent.

c) We will be happy to propose new functionality for SPC-3 in the event that this Public Review comment is rejected, as below.

SPC-3 PROPOSALS

VERITAS would like to get some feedback on the approach to be taken to the inclusion of new definitions in SPC-3 to address the issue of a reservation being removed when initiator that made the reservation deregisters. In line with out Public Review comment, the two possibilities are:

1) Define two new Persistent reservation type codes (7h and 8h) as in Table 1 below;

2) Define two new Persistent reservation type codes (7h and 8h) as in Table 1 below, and mark codes 5h and 6h as obsolete

In addition, VERITAS has found another approach which will we believe will provide the desired functionality, as follows:

The sentence in subclause 5.5.3.6.1 that was the focus of the VERITAS Public Review comment, namely:

Any persistent reservation associated with that unregistered initiator shall be released.

requires a device server to keep track of the Initiator that created the reservation. However the device server has no way to report that information in response to a query. A new service action for the PERSISTENT RESERVE IN command, READ KEYS PLUS INITIATOR, could be defined which could return an Initiator identifier in addition to the information currently contained in READ KEYS, as in Table 2 below. Using such a service action would obviate the need for an Initiator undergoing a graceful shutdown to removes its registration, as when the Initiator is reinitialized it could detect the existence of an existing registration, and not submit another registration.

.

Table 1 — Persistent reservation type codes

Code	Name	Description
0h		Obsolete
1h	Write Exclusive	**Reads Shared**: Any application client on any initiator may initiate tasks that request transfers from the storage medium or cache of the logical unit to the initiator. **Writes Exclusive**: Any task from any initiator other than the initiator holding the persistent reservation that requests a transfer from the initiator to the storage medium or cache of the logical unit shall be terminated with RESERVATION CONFLICT status.
2h		Obsolete
3h	Exclusive Access	**Reads Exclusive**: Any task from any initiator other than the initiator holding the persistent reservation that requests a transfer from the storage medium or cache of the logical unit to the initiator shall be terminated with RESERVATION CONFLICT status. **Writes Exclusive**: Any task from any initiator other than the initiator holding the persistent reservation that requests a transfer from the initiator to the storage medium or cache of the logical unit shall be terminated with RESERVATION CONFLICT status.
4h		Obsolete
5h	Write Exclusive – Registrants Only	**Reads Shared**: Any application client on any initiator may initiate tasks that request transfers from the storage medium or cache of the logical unit to the initiator. **Writes Exclusive:** A task that requests a transfer to the storage medium or cache of the logical unit from an initiator that is not currently registered with the device server shall be terminated with RESERVATION CONFLICT status.
6h	Exclusive Access – Registrants Only	**Reads Exclusive**: A task that requests a transfer from the storage medium or cache of the logical unit to an initiator that is not currently registered with the device server shall be terminated with RESERVATION CONFLICT status. **Writes Exclusive:** A task that requests a transfer to the storage medium or cache of the logical unit from an initiator that is not currently registered with the device server shall be terminated with RESERVATION CONFLICT status.
7h	Write Exclusive – All Registrants	as in 5h above, with the exception that when the Initiator that created the reservation deregisters, the reservation is not released (see 5.5.3.6.1)
8h	Exclusive Access – All Registrants	as in 6h above, with the exception that when the Initiator that created the reservation deregisters, the reservation is not released (see 5.5.3.6.1)
9h - Fh	Reserved	

Table 2 — PERSISTENT RESERVE IN parameter data for READ KEYS+INITIATOR

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)			GENERATION				
3								(LSB)
4	(MSB)			ADDITIONAL LENGTH (n-7)				
7								(LSB)
				Reservation key & initiator list				
8	(MSB)			First reservation key				
15								(LSB)
16	(MSB)			First initiator identifier				
31								(LSB)
				. . .				
n-23	(MSB)			Last reservation key				
n-16								(LSB)
n-15	(MSB)			Last initiator identifier				
n								(LSB)