T11/02-483r0

Date: November 6, 2002

To: T10 Committee (SCSI)

From: Roger Cummings (VERITAS)

Subject: Further Work on Migration to Persistent Reservations

VERITAS is continuing to work on the migration of our products towards supporting Persistent Reservations instead of (original) Reservations. As a result of that work, we have identified two further areas in which we feel that the definition of persistent reservations need to be amended or enhanced. This document defines the problems that have led us to this conclusion, and outlines one more possible solutions. We would like to present this to the CAP Working Group at this time to see if the problems that we describe are common throughout the industry, and to obtain some guidance with regards to possible solutions. As a result of the discussion on this document, a number of proposals for detailed changes to SPC-3 will be created if and as appropriate.

The two areas that we have identified are:

1) The need for an analog to the third-party function supported by original reservations;

2) The need to be able to mix usage of reserve/release and persistent reservations when dealing with tapes.

The background to each of these areas is described below, along with some suggestions for enhancements to the persistent reservation definitions to solve the problems.

THIRD PARTY PERSISTENT RESERVATIONS

The definitions of RESERVE (10) and RELEASE (10) in SPC-2 include the concepts of Third-Party Reserve and Release and Superseding Reservations. These concepts were specifically created for use in the situation where multiple Initiators and a ‘copy engine” which implements the EXTENDED COPY command are present in a system, and they are used extensively in that situation by the VERITAS Data Protection products.

The specific usage of these concepts is as follows. Note that this same sequence will be performed with both Logical Units (source and target) which the copy engine will access, but only one Logical Unit is shown for simplicity

1) Host A sends a Reserve to Logical Unit 1.

2) Host A accesses and verifies the data on Logical Unit 1 (for a serial access Logical Unit some media loading & positioning commands may also be necessary).

3) Host A verifies that Copy Engine C has no outstanding operations.

4) Host A sends a third-party (superseding) Reserve to Logical Unit 1 containing the address of Copy Engine C.

5) When the Reserve succeeds, Host A sends one or more Extended Copy commands to Copy Engine C which cause it to access Logical Unit 1.

6) When the Extended Copy commands are complete, Host A sends a non third-party (superseding) Reserve to Logical Unit 1.

7) Host A accesses Logical Unit 1 if necessary (for a serial access Logical Unit some media positioning and media unloading commands may also be necessary).

8) Host A sends a Release to Logical Unit 1.

There are three key aspects to the above sequence:

a) At no time is more than one Initiator Port able to access the Logical Unit;

b) Once step 1) is complete, there is no opportunity for another host or copy engine to interject in the process until step 8 is completed;

c) The Copy Engine does not need to be aware of the Reservation status at all.

None of these three aspects are directly supported by the present definition of Persistent Reservations.

a) A group reservation would allow both Host A and Copy Engine C to access the Logical Unit at the same time (and any other Initiator could register and also gain access);

b) Another Initiator could also issue a Preempt and gain access between steps 1 & 8;

c) The Copy Engine is required to Register with the Logical Unit before any type of Persistent Reservation access can be granted.

The equivalent sequence using Persistent Reservations would need to be something similar to:

1) Host A Sends a Register and Ignore Key command followed by a an Exclusive Access Reserve to Logical Unit 1.

2) Host A accesses and verifies the data on Logical Unit 1 (for a serial access Logical Unit some media loading & positioning commands may also be necessary).

3) Host A verifies that Copy Engine C has no outstanding operations, and sends an Extended Copy command which instructs Copy Engine C to Register with Logical Unit 1. (This needs to be a new type of Register which confers no access on the Copy Engine even if a Registrants Only or All Registrants reservation is established - so that this process fails in the case of a Preempt by another Initiator).

4) Host A sends a (new) Service Action to Logical Unit 1 which Transfers the Exclusive Access reservation to the key value registered earlier by Copy Engine C, but does not remove Host A's registration.

5) When the Transfer succeeds, Host A sends one or more Extended Copy commands to Copy Engine C which cause it to access Logical Unit 1.

6) When the Extended Copy commands are complete, Host A sends (new) Service Action to Logical Unit 1 which Transfers the Exclusive Access reservation back to its registered key value (and this service action will only be accepted from the same Initiator port that issued the transfer in step 4). It also sends an Extended Copy command which instructs Copy Engine C to Register with Logical Unit 1 with a key of zero (unless the new type of Register in step 3 was used).

7) Host A accesses Logical Unit 1 if necessary (for a serial access Logical Unit some media positioning and media unloading commands may also be necessary).

8) Host A sends a Release to Logical Unit 1.

The above sequence requires two new features to be define for Persistent Reservations:

a) A new type of Registration which does not participate in Registrants Only or All Registrants reservations.

b) A new Persistent Reserve Out Service Action which can Transfer a Write-Exclusive or Exclusive Access reservation between two registered Initiator Ports without any possibility of another Initiator port being able to influence the reservation.

MIXING (ORIGINAL) RESERVATIONS AND PERSISTENT RESERVATIONS WITH TAPES

A number of server platforms issue a Reserve Command to a serial access device as a matter of course when opening a Volume. The Reserve is issued by the operating system itself, or by the tape class driver, independently of any application. In some, but not all, cases this behavior is configurable, but there is considerable resistance from users to changing the configuration, and it cannot in any event be done on a per-application basis.

Clearly, this behavior makes use of persistent Reservations with tape drives impractical. If an application issues a Persistent Reserve to the Tape LUN before opening a Volume, then the open will fail. If the application attempts to issue a Persistent Reserve after the platform has issued a Reserve, a Reservation Conflict will be returned, but this could be a result of another platform having an Exclusive access Persistent Reserve or this same platform having issued a Reserve.

The VERITAS Data Protection applications are required to operate in a heterogeneous configurations containing mature server platforms which are no longer being developed, but which exhibit the above behavior. Until those platforms are removed from the supported list, therefore, there is little prospect of our applications supporting persistent reservations with tape devices, unless the rules for mixing Reservations and Persistent Reservations are changed. Any such change will inevitably increase the complexity of those rules from the “a Reservations causes Persistent Reservations to be rejected, even one Registration causes Reservations to be rejected” situation that exists today.

If changes to the “mixing” rules can be considered, the following are proposed for consideration:

1) If a Reserve is issued from an Initiator Port which has access to a Logical Unit via an existing Persistent Reservation, then that Reserve is allowed but no action is performed.

2) The receipt of a Reserve by a Logical Unit which supports Persistent Reservations causes the synthesis in that LU of a Registration for that Initiator Port (if no registration exists) with a defined key value (suggested FFFFFFFF), and an Exclusive Access Reservation if no reservation already exists. If a previous reservation does exist, then if the reservation type is that the Initiator Port is allowed access, the Reserve succeeds, otherwise a Reservation Conflict is generated. The receipt of a Release by a Logical Unit with an existing synthesized reservation will then be treated as the equivalent of a Register with a zero key value, and both the Registration and Reservation will be removed.