Date: July 2, 2003

To: T10 Committee (SCSI)

From: Roger Cummings (VERITAS)

Subject: Two Persistent Reservations problems - latest information

1.0 INTRODUCTION

In May 2003 I presented (on behalf of VERITAS) an update on two problems with the Persistent Reservations definitions that existed in SPC-3 that I had originally presented in November 2002. These were problems that we felt needed to be addressed in order to be able to migrate applications and operating systems away from supporting (original) Reserve and Release and towards Persistent Reservations. These two problems were:

1) The need to be able to mix usage of reserve/release and persistent reservations in the same system & storage network when dealing with tapes;

2) The need for an analog to the third-party function supported by original reservations in Persistent Reservations.

Document 02-483r1 documents a background to those problems, and outlined some proposed solutions. These solutions were discussed at the May CAP meeting, and I received some direction from the meeting as to how the solutions should be developed. I have attempted to follow that direction, but further work at VERITAS has generated some new information related to these problems that we feel needs to be addressed. That information is documented below. Detailed proposals on solutions to the problems that take this information into account are given in separate documents as referenced below.

2.0 LATEST INFORMATION

2.1 Mixing Reservations & Persistent Reservations with Tapes

The May CAP meeting suggested defining a new command that converts a Reserve/Release reservation to a Persistent Reservation. It was noted that a solution that accounts for error recovery issues probably will require adding text defining a new behavior for the RELEASE command in SPC-3.

VERITAS has unfortunately discovered that such a conversion is insufficient to solve this problem. Our investigation discovered that there are certain operating systems that are profligate with the use of reserves and releases, that is the issue many of each type during the execution of specific code paths.

Because there is no way to test that a reservation still exists, and because writing to an unreserved device has to be protected against, many platforms take the approach of issuing multiple Reserve commands during the mounting of a tape volume. Issuing a subsequent Reserve command to an already-reserved device has no detrimental effect, and successful status is returned. The author has been give a demonstration where a total of 26 reserves were issued during a tape mount by a combination of an application, the OS platform, and the device driver.

By the same logic, some platforms also issue multiple Releases during an unmount process. A different platform to that described above has been known to issue 10 Releases. As an aside, much legacy

code has apparently been written on the assumption that a Release never fails, and the SPC-2 definition which defines that Release shall obtain a reservation conflict in the situation where a logical unit has executed a PERSISTENT RESERVE OUT command with one of the registration service actions and is still registered by any initiator, has generated considerable concern.

Other strange sequences in which Reserves and Releases are used with multiple rewinds as well as forward space block to non-existent locations have also be seen in such circumstances! In some cases FCP Process Logouts and Logins are also incorporated in these sequences.

From the above work, it has become clear that a new command which converts a Reservation to a Persistent Reservation is insufficient to solve the problem. Once the conversion command is issued and completely, it is to be expected that further Reserve commands will be issued against that same Logical Unit. It is essential that these commands do not fail, or the mount process will be aborted.

Furthermore, we have discovered that some OS platforms do not permit an application to have direct access to a logical unit (via a passthru interface or some other special construct), at the same time as the storage stack is accessing the device. Thus if the application issues the new “conversion” command, and then passes control to the storage stack, it cannot regain access to issue further commands until the stack passes back control, which is often only done at specific points when the device is quiesced.

As a result of all of the above, we believe that there is no alternative to creating a situation by which Reserve and Release commands that are received from an Initiator Port that has access under a pre-existing Persistent Reservation are accepted, but have no effect on the state of the Persistent Reservation.

We recognize that this will require adding text defining a new behavior for the RESERVE command in SPC-3 in addition to that for the RELEASE command mentioned @ the May CAP meeting, but contend that there is no alternative if this problem is to be solved.

Document 03-232r0 contains a proposal for changes to the definitions of Persistent Reserve In and Persistent Reserve Out to define an Modified Reservation Handling (MRH) feature to create this situation.

In addition, further work has convinced us that defining a conversion command is not necessary for our applications. In the small number of situations where a Reserve is in place when we try and issue a Persistent Reserve, there are existing features that we can use to overcome that Reserve. However, if the CAP group still feels that such a conversion command is necessary for a general solution, we will be pleased to create a definition for such a command in a proposal to the September CAP group.

2.2 Third Party Persistent Reservations

The May CAP meeting recommended defining a Move Reservation PERSISTENT RESERVE OUT service action to address the Third Party Persistent Reservation problem.

That recommendation is viable and Document 03-233r0 contains a proposal to define such a service action. However we have discovered an additional problem that we believe will have to be addressed in order for Persistent Reservations to completely replace Reservations, and we believe that our proposed solution to that problem may be of more general interest and also able to address this problem. The additional problem is described below, and Document 03-xxxr0 a proposal to address both problems.

2.3 Additional Persistent Reservations problem

A small number of platforms support multiple paths to a tape device in a manner that is managed entirely by the lower layers of a storage stack and is not visible to the higher layers or any application - even via a passthru interface. In this model, switching between paths is done automatically in

response to various events, and apparently also for load-balancing purposes. Reservations are handled in such a scheme by having a retry process in response to a “Reservation Conflict” status on each path which issues a Target Reset to remove the existing reservation and reestablishes the Reservation for the new path. Clearly the establishment of a Persistent Reservation will prevent this happening, and cause the path switchover to fail.

The Registrants Only or All Registrants Persistent Reservations types could be used to address this problem, except that would also allow a device not related to the path group to Register and gain access, and that is unacceptable for some classes of application.

What is needed is therefore a type of reservation which allows ONLY a specific group of paths to be defined as having access, with each path being added incrementally as it is activated - as it may not be known ahead of time.

Document 03-234r0 contains a proposal to address this need. It modifies the behavior of the Registrants Only or All Registrants Persistent Reservations types such that when a new feature is activated, only those registrants that are explicitly JOINed to the reservation have access under it, rather than any existing or future Registrant.