April 27, 1999

To: T10 Committee

From: Tom Coughlan
Compaq Computer Corporation
Mail Stop ZKO3-4/U14
110 Spitbrook Road
Nashua, New Hampshire
Telephone: 603-884-0933
E-mail: tom.coughlan@compaq.com

Subject: Still more on Persistent Reservation (99-182r0)

## Change I. Allow the Device Server to ignore the existing key for Register service actions

The current revision of SPC-2 (Rev. 9) requires that an initiator with a previously established reservation key must know the key value before it can perform a Register service action. This requirement is not necessary, and it creates a severe difficulty in SAN environments, where the user may wish to alternately configure the storage device in multiple operating system environments.

The difficulty arises because different operating system types, operating system modes (for example, clustered and non-clustered), and operating environments such as boot ROM code, can not be expected to know each other's key assignment algorithm. As a result, when a storage device that was being used with persistent reservations in one environment is moved to another environment, it is extremely difficult for the initiator to register, and clear the leftover state from the old environment.

The argument in favor of the status quo is presumably based on a desire to protect the logical unit from an initiator that is not participating in the proper key assignment algorithm. This type of security is not one of the capabilities provided by persistent reservations. In this case, the intended protection is defeated by the fact that each operating system and boot ROM environment must implement code to deal with the possibility that a storage device may contain old persistent reservation state, leftover from a previous environment. With the current Persistent Reservation specification, this code must attempt a Registration with a key of zero, and if a failure occurs, it must issue a Read Keys action, and then attempt to Register with each of the keys in the returned list. One of these keys should work, unless the initiator has been preempted in the meantime. Thus, if none of the keys work, the initiator must try again with a key of zero. Once the initiator is finally registered, it can clean up the leftover state.

This code adds unnecessary complexity and delay. The proposed change is to allow the device server to ignore the Reservation Key field for Register service actions.

## *Detailed Change:*

### 5.3.2.3 Registering

To establish a persistent reservation the initiator shall first register with a logical unit.

~~If the initiator has not yet established a reservation key or the reservation key has been removed,~~ ~~t~~The registration is accomplished by issuing a PERSISTENT RESERVE OUT command with service action of REGISTER with the following parameters:

a) APTPL bit optionally set to one; and
~~b)reservation key set to zero; and~~
~~c)~~b) service action reservation key set to a non-zero value.

~~If the initiator has an established registration it may change its reservation key. This is accomplished by issuing a PERSISTENT RESERVE OUT command with service action of REGISTER with the following parameters:~~

~~a)APTPL bit optionally set to one;~~
~~b)reservation key set to the value of the previously established reservation key; and~~
~~c)service action reservation key set to a non-zero value.~~

If a PERSISTENT RESERVE OUT with a REGISTER service action is attempted, but there are insufficient device server resources to complete the operation, the device server shall return a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense data shall be set to INSUFFICIENT REGISTRATION RESOURCES.

NOTE 3 It is recommended a target have enough resources to handle a registration from each initiator known to the target.

In response to a PERSISTENT RESERVE OUT with a REGISTER service action the device server shall perform a registration by doing the following as an uninterrupted series of actions:

a) Process the registration request regardless of any persistent reservations;
b) process the APTPL bit;
c) ignore the contents of the RESERVATION KEY, SCOPE and TYPE fields;
d) map the reservation key to the registering initiator using the initiator identification and, if available, the initiator port's world wide identification;
e) register the reservation key without changing any a persistent reservation that may exist; and
f) retain the reservation key and associated information.

### 7.12.2 PERSISTENT RESERVE OUT parameter list

**…**

The RESERVATION KEY field contains an 8-byte value provided by the application client to the device server to identify the initiator that is the source of the PERSISTENT RESERVE OUT command. The device server shall verify that the RESERVATION KEY field in a PERSISTENT RESERVE OUT command matches the registered reservation key for the initiator from which the task was received, except for the REGISTER service action. The device server shall ignore the RESERVATION KEY field for the REGISTER service action. ~~for an unregistered initiator which shall~~

~~have a reservation key value of zero. If a PERSISTENT RESERVE OUT command~~ If one of the other service actions specifies a RESERVATION KEY field other than the reservation key registered for the initiator, the device server shall return a RESERVATION CONFLICT status. The reservation key of the initiator shall be verified to be correct regardless of the ~~SERVICE ACTION and~~ SCOPE field value~~s~~. The obsolete field in Bytes 22 and 23 was defined in a previous standard.

## Change II. The reservation established by Preempt and Abort must apply to tasks that arrive after the abort.

The Preempt and Abort service action is used by cluster software, to provide a way for a cluster to remove a member that is no longer communicating with the remaining cluster members. In order for a cluster to safely remove a member, the remaining members must ensure that there are no queued tasks associated with the removed member at the shared logical unit, and that future attempts by the removed member to queue certain tasks to the logical unit are blocked. The Preempt and Abort service action achieves this goal, by aborting commands, clearing other state, and establishing a reservation that blocks commands from the specified initiator(s).

It is essential that the Preempt and Abort be performed as an uninterrupted series of actions, as currently specified in the standard. Unfortunately, the standard also contains a statement that introduces some doubt about the meaning of "uninterrupted". In the clause that describes the abort function, the standard says:

> After GOOD status has been returned for the PERSISTENT RESERVE OUT command, new tasks are subject to the persistent reservation restrictions established by the preempting initiator

This statement may be interpreted to mean that new tasks may to enter the task set after the device server has executed the abort, but before GOOD status is returned, without being subjected to the new reservation. This window creates the possibility that there will be tasks in the task set from the preempted initiator when the Preempt and Abort completes. This can lead to undetected data corruption in a cluster.

The proposed change is to make this sentence an explicit requirement that new tasks from the preempted initiators shall be subject to the new reservation.

### *Detailed Change:*

#### 5.3.2.5.3 Preempting an existing persistent reservation with the PREEMPT AND ABORT service action

The initiator's request for and the device server's responses to a PERSISTENT RESERVE OUT command PREEMPT AND ABORT service action are identical to the PREEMPT service action (see 5.3.2.5.2) except for the following additions. If no reservation conflict occurred, the device server shall do the following as part of the uninterrupted series of actions:

a) Every task from all preempted initiators shall be terminated as if an ABORT TASK SET task management function had been performed by each of the preempted initiators. ~~After GOOD status has been returned for the PERSISTENT RESERVE OUT command,~~ All new tasks from the preempted initiators are subject to the persistent reservation restrictions established by the preempting initiator;
b) The device server shall clear any ACA or CA condition associated with an initiator being preempted and shall clear any tasks with an ACA attribute from that initiator. If TST=000b (see 8.3.4), then ACA or CA conditions for initiators, other than the initiator being preempted, shall prevent the execution of the PERSISTENT RESERVE OUT command that shall end with a status of ACA ACTIVE if NACA=1 (see SAM-2) or BUSY if NACA=0. If TST=001b, then ACA or CA conditions for initiators other than the initiator being preempted shall not prevent the execution of the PERSISTENT RESERVE OUT command; and
c) For SCSI devices that implement the PREVENT ALLOW MEDIUM REMOVAL command, the device server shall perform an action equivalent to the execution of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field equal to zero for the initiator or initiators being preempted (see 7.13).

## Change III. Clarification of Preempt and Abort behavior when there is no existing reservation.

The current revision of SPC-2 (Rev. 9) specifies the behavior of Preempt and Abort by providing a reference to the behavior of the Preempt service action. The description of Preempt describes two cases: a) preempting reservations, and b) preempting registrations.

The purpose of this change is to provide an explicit statement that the abort part of the Preempt and Abort shall occur in both of the two cases listed above. Although this fact can be inferred from a careful reading of the existing text, we believe that an explicit statement is needed. The reason is because it may be counter-intuitive that simply preempting an initiator's *registration* also causes an abort of all the initiator's queued commands.

### *Detailed Change:*

Insert the following sentence before the last sentence of the section:

**5.3.2.5.3 Preempting an existing persistent reservation with the PREEMPT AND ABORT service action**

The actions described above shall be performed for all initiators that are registered with the SERVICE ACTION RESERVATION KEY, independently of whether the registered initiator(s) hold the reservation.