Accredited Standards Committee

NCITS, National Committee for Information Technology Standards

Doc: T10/97-218r2
Date: 9/18/97
Project: SPC-x
Ref Doc:
Reply to: Bob Snively

To: T10 Membership

From: Bob Snively

Subject: Review of Persistent Reservation

During review of persistent reservation, three issues were identified which should be resolved. Those issues and proposed solutions are discussed in revision 0 of this document. Revision 1 includes the following modifications to the original proposal.

Conditions for performing persistent reservation operation:

Rewrites the proposed text with more direct logic.

Proposes a relevant change to the reservation overview section in SPC.

1.0 Conditions for performing persistent reservation operation

1.1 Overview

There is a phrase in the discussion about creating reservation conflicts with the PERSISTENT RESERVE OUT command in section 7.13 that has profound implications.

> Commands from any initiator that conflict with a successfully established persistent reservation shall be rejected with a status of RESERVATION CONFLICT. The following commands shall not conflict with a reservation established by the PERSISTENT RESERVE OUT command:
>
> PERSISTENT RESERVE IN
>
> PERSISTENT RESERVE OUT (with an Service action of Preempt)
>
> PERSISTENT RESERVE OUT (with an Service action of Preempt and Clear)
>
> PERSISTENT RESERVE OUT (with a Reserve service action that does not conflict with established persistent reservations or tasks)

The offending words here are “or tasks”. That implies that a PERSISTENT RESERVE OUT command with a service action of Reserve can only be executed after all queued tasks that may conflict with the new reservation are completed. The words apparently apply to all tasks, including those

that are enabled and are actually being performed as well as those that are blocked or queued. It even includes tasks that are queued after an ordered PERSISTENT RESERVE OUT command.

Section 5.3 further describes the behavior using the following text. This sentence establishes the context for the entire paragraph.

> For commands that do not explicitly read or write the medium, the applicable reservation restrictions depend solely on the type of reservations that are established at the time the command reaches the device server.

Section 5.3.1 further describes the behavior using the following text. This text is in the context of defining the conditions for a reservation conflict, independent of the type of task being examined.

> The device server shall test for reservation conflicts at the time when a task enters the enabled task state. If a reservation conflict precludes any part of the command, none of the command shall be performed.

Those two sections really do not help clarify the behavior of reservations in a queued environment, and they partially conflict with each other. In addition, they leave some questions with respect to the execution of the relationship between PERSISTENT RESERVATION IN/OUT commands.

Note 26 (below) seems to imply the behavior I would actually expect. That is, when a PERSISTENT RESERVE OUT conflicts with a task that has already been passed to the device or is passed to the device before the PERSISTENT RESERVE OUT command is enabled and begins execution, the occurrence of a RESERVATION CONFLICT status and the time of effectiveness of a reserve service action is vendor specific. Notes 27 and 29 provide similar warnings for Preempt and Preempt & Clear.

> NOTE 26: For the simplest predictable behavior, the Reserve service action should be performed with the Ordered task attribute.

1.2 Suggested Change

At the earliest time allowed by the standards process:

1) I would suggest that we remove the words “or tasks” from the document. That would remove any implication that the presence of conflicting tasks would cause a reservation conflict response to a reservation command. In addition, the section should be rewritten as follows to clarify the meaning. The rewritten text removes any reference to PERSISTENT RESERVE OUT with a service action of Reserve because that command follows the rule specified in the first sentence of the modified text. Note that reservation conflicts caused by reservation key mismatches are addressed in 7.13.2 and need not be addressed in this paragraph.

> Commands from any initiator that conflict with a successfully established persistent reservation shall be rejected with a status of RESERVATION CONFLICT with the following exceptions
>
> > PERSISTENT RESERVE IN shall not conflict with any persistent reservation.
> >
> > PERSISTENT RESERVE OUT with a service action of Preempt shall not conflict with any persistent reservation.
> >
> > PERSISTENT RESERVE OUT with a service action of Preempt and Clear shall not

conflict with any persistent reservation.

2) I would suggest that the text of the model be modified to clarify the queuing requirements and to incorporate the intent of NOTES 26, 27, and 29. The last three paragraphs of 5.3 and section 5.3.1 should be rewritten as follows. Section 5.3.1 should be included within section 5.3, which renumbers paragraphs 5.3.2 and 5.3.3 to 5.3.1 and 5.3.2 respectively.

Certain commands that do not explicitly read or write the medium may conflict with the restrictions established by a previous reservation. If a reservation conflict precludes the execution of any part of the command, none of the command shall be performed and the device server shall terminate that command with a RESERVATION CONFLICT status. The command may be checked for reservation conflicts at any time between the time the command is delivered to the device server and the time the device server begins to execute the command. The particular reservation restrictions imposed are highly dependent on the relationship between the command and the type of reservations. The restrictions do not depend on the reservation management method and may not be the same as the restrictions for commands that explicitly read or write the medium. The reservation restrictions for commands that do not explicitly access the medium are defined in the device model clause or in the clause defining that specific command.

The reservation restrictions for commands that manage reservations may conflict with previous reservations. The clauses describing the reservations management commands define the interactions between the commands and previously established reservations. (See 7.12, 7.13, 7.18, 7.17, 7.22, and 7.21.)

Commands that explicitly read or write the medium may conflict with the restrictions established by a previous reservation. If a reservation conflict precludes the execution of any part of the command, none of the command shall be performed and the device server shall terminate that command with a RESERVATION CONFLICT status. The command may be checked for reservation conflicts at any time between the time the command is delivered to the device server and the time the device server begins to modify the data on the medium or deliver data from the medium. For each command, this standard or a related command standard (see 3.1.11) defines the conditions that result in RESERVATION CONFLICT. The conditions are identified as part of the device model or command definition.

The time of effectiveness of a reservation with respect to other tasks being managed by the device server is vendor specific. Successful completion of a reservation command indicates that the new reservation is active. An active reservation may apply to some or all tasks queued before the completion of the reservation command. The reservation shall apply to all tasks received after successful completion of the reservation command. The execution of a reservation command shall be performed as a single event, such that no other command present in the queue shall be uncertain of the presence of a reservation at the time the command tests for the presence of a reservation.

Because a device server is unable to differentiate among the reservations made by different application clients running on an initiator, all application clients on the initiator have the same access restrictions. When multiple application clients are accessing a single device server from one initiator, the application clients should coordinate reservations and persistent res-

ervations.

3) I recommend that notes 26, 27, and 29 be removed, since the information is now present in the reservation model.

2.0 Duplicate reservations

2.1 Overview

Non-duplicate reservations are simple for a target to manage. It is easy enforce the rule that each must be cleared separately. The text of 7.13.1.2 makes this very clear:

> New persistent reservations that do not conflict with an existing persistent reservation shall be executed normally. The persistent reservation of a logical unit or the persistent reservation of extents having the same type value shall be permitted if no conflicting persistent reservations are held by another initiator. When such overlapping persistent reservations are released, each of the extent reservations and the logical unit reservation shall be removed with a separate Release service action.

However, two identical reservations from the host have a less clear behavior. In particular, when a release comes through, should the first, the second, or both be cleared? I expect that in many implementations, there may not even be a simple mechanism to count and compare the number of executions of identical Reserve service actions and Release service actions.

2.2 Suggested Change

At the earliest opportunity in the standardization process, I suggest we make the following changes.

I would propose that the paragraph be expanded to allow a Release service action to release all identical reservations. With that addition, it will not matter if individual copies of the reservations are kept or not. It will also not matter how many Release service actions occurred relative to the number of Reserve service actions. Text is already present to indicate that the release of reservations that do not exist is not an error. The new text would read:

> New persistent reservations that do not conflict with an existing persistent reservation shall be executed normally. The persistent reservation of a logical unit or the persistent reservation of extents having the same type value shall be permitted if no conflicting persistent reservations are held by another initiator. When such overlapping persistent reservations are released, each of the extent reservations and the logical unit reservation shall be removed with a separate Release service action. Multiple identical reservations from the same initiator are all simultaneously released by a single Release service action that matches the reservations.

Note that there is one peculiar result of this. If you have independent control processes operating against the same LU from the same initiator, then one of those control processes may be very much surprised to find that its reservations have been removed by another control process. That is probably bad program design, rather than a SCSI architecture error.

3.0 Reservation key clearing

3.1 Overview

At present, text in 7.13.1.1 says:

> For each initiator that performs a PERSISTENT RESERVE OUT Register service action, the device server shall retain the reservation key until the key is changed by a new PERSISTENT RESERVE OUT command with the Register service action from the same initiator or until the key is reset to the default value of zero by powering down the logical unit, if the last APTPL received by the device server was zero (see 7.13.2) or by performing a Clear, Preempt, or Preempt and Clear service action.

Similar text is found in

> 7.13.1.3, Release, item a)
>
> 7.13.1.4, Clear, first paragraph
>
> 7.13.1.5, Preempt, 4th paragraph
>
> 7.13.1.6, Preempt and Clear, 7th paragraph
>
> 7.13.2, PERSISTENT RESERVE OUT parameter list, 6th paragraph

It is never explicitly indicated what “reset to the default value of zero” really means. I believe that it was intended to mean that the reservation key is unregistered and disassociated from any relationship to an initiator port. At the same time, any new registration to the device from the unregistered initiator would use the default reservation key of zero. That is a necessary behavior to properly handle the proposed “Write Exclusive, Registrants Only” reservation type. If the state of being registered is not cleared by the action of resetting the reservation key, then an offending initiator cannot be locked out. In addition, extinct initiator tables cannot be recovered, but must remain available for setting to a non-default value.

3.2 Suggested Change

I suggest that the following changes be installed in the document at the earliest time allowed by the standardization process.

I suggest that the text be more specific in its definition of resetting the reservation, explicitly indicating that the initiator becomes unregistered and any resources associated with the initiator be released. The demonstration paragraph above (7.13.1.1) would be changed to read:

> For each initiator that performs a PERSISTENT RESERVE OUT Register service action, the device server shall retain the reservation key until the key is changed by a new PERSISTENT RESERVE OUT command with the Register service action from the same initiator or until the initiator registration is removed by one of the following actions:
>
> 1) powering down the logical unit, if the last APTPL received by the device server was zero (see 7.13.2)
>
> 2) performing a Clear service action
>
> 3) performing a Preempt service action

4) performing a Preempt and Clear service action

When the reservation has been removed, the default reservation key of zero shall be used when the initiator registers a reservation key again. When the reservation has been removed, no information is reported for the initiator in the Report Keys service action.

Similar text indicating that “the initiator registration is removed” would be installed in the following locations:

7.13.1.3, Release, item a)

7.13.1.4, Clear, first paragraph

7.13.1.5, Preempt, 4th paragraph

7.13.1.6, Preempt and Clear, 7th paragraph

7.13.2, PERSISTENT RESERVE OUT parameter list, 6th paragraph

Sincerely,

Robert Snively

Phone:(415) 786-6694

Email: bob.snively@sun.com