To:	INCITS T10 Committee
From:	Paul Entzel, Quantum
Date:	23 June 2006
Document:	T10/06-261r0
Subject:	ADC-2: Fix bridging reservations and other stuff

1 Revision History

Revision 0:
Posted to the T10 web site on 23 June 2006.

2 Reference

T10/ADC-2 revision 5a.

3 General

This proposal addresses an action item assigned to me at the 8 May 2006 working ADI group meeting (see T10/06-228 action item 6-38). I also used this proposal to address a couple of other issues that have been brought to my attention in the last couple of weeks.

Proposed additions to the ADC-2 standard are shown in blue text; proposed deletions are shown in ~~red crossed out text.~~

3.1 Bridging reservation problem

Subclause 4.2.3.2 in ADC-2 states that the local SMC device server shall process reservation and persistent reservation commands that it receives and shall not pass them through to the remote SMC device server. The justification behind this requirement is that only the local SMC device server can distinguish the true I_T nexus which is required to support the reservation. The Remote SMC device server can only see the I_T nexus consisting of the ports that establish the link between the DT device and the library, typically a pair of ADT ports.

A problem arises when the library enables multiple ports to access the same device server and at least one of them is a bridged port. In other words, the library enables bridging in more than one DT device to a single remote SMC device server. Or, the library enables bridging in a DT device to a remote SMC device server that can also be accessed via a primary port in the library. In the first case, the reservations are being managed by two or more distinct local SMC device servers without any interaction between the device servers. In the second case, the reservations are being managed by the local SMC device server and the remote SMC device server without enough shared information. This is not a unique problem, it is common to all SCSI protocol bridges.

Classic reservations (RESERVE and RELEASE) would not be difficult to support this distributed reservation management for several reasons:

1. Reservations are exclusive to a single I_T nexus.
2. There is no mechanism for report reservation status except by trying a command and see if it works.
3. Reservations are a single step process versus the two step process of persistent reservations.

If all we needed to make work were classic reservations we could fix the problem by simply passing the RESERVE and RELEASE commands through to the remote SMC device server. The local SMC device server would still need to distinguish which I_T nexus that access it has reservation rights, but all other local SMC device servers or direct access port would be denied access by the reservation established in the remote SMC device server.

Persistent reservations are a whole different story. The fact the persistent reservations work the exact opposite as classic reservations in each item in the list above is what complicates them. Let's look at them one at a time.

3.1.1 Reservations are exclusive to a single I_T nexus

With persistent reservations, some are exclusive, most are not. To deal with this case we would need to require the local SMC device server to report any persistent reservation established via a primary port to the remote SMC device server. After that, we have two choices:

1. If we chose to continue the practice of honoring the reservations in the local SMC devices servers where the true I_T nexus is known, we would also need to invent a method for a remote SMC device server to report the establishment of a persistent reservation to the local SMC device servers. The local SMC device servers could then honor the persistent reservation.
2. If we chose to move the management of reservations to the remote SMC device server then we will need a method of indicating the I_T nexus that is the source for every command. Any volunteers?

3.1.2 There is no mechanism for report reservation status except by trying a command and see if it works

Not so with persistent reservations. You can request a list of all the registered keys. This list includes all reservation keys registered with the device server over all target ports, and includes multiple entries for duplicate keys that have been registered. You can request information about the current reservation that includes the key, type, and scope. And, you can request a full report about the reservation that has target port and initiator port information in it also. Of course, we don't have to support all of these reports…

3.1.3 Reservations are a single step process versus the two step process of PRs

Persistent reservations are a two steps, first register a key, then reserve a logical unit. The standard allows only a single reservation key for a given I_T nexus. Let's look at a scenario with to initiators:

1. Initiator A registers with key 1234.
2. Initiator B registers with key 5678.

If the bridging manager passed the first registration on to the Remote SMC device server, the second registration needs to be held back or it will over-ride the first registration. If initiator B then attempts to establish a PR with the SMC device server, the bridging manager must either establish the reservation with the wrong key, or first change the key then establish the reservation.

Another alternative would be to not register with the Remote SMC device server until an initiator attempts to establish a persistent reservation, then do both the register and reserve operations while processing the PR OUT that attempts to reserve the logical unit. This has the problem of leaving the device server free to accept classic reservations from another port while a registration is in place.

3.1.4 Conclusion

Solving this problem by providing a solution that allows multiple ports to access a single Remote SMC device server where one or more of the ports is a DT device bridging commands is proving to be more trouble than it is worth. The proposed solution is to forbid such a configuration and provide a method for how a bridging manager to enforce this solution. See section 4 for the proposed change.

3.2 "requests" versus "commands"

It has been pointed out to me by an engineer attempting to implement ADC in a library that the standard is a little loose on the use of the term "request". Actually, that was my conclusion, the question I received related to this paragraph in subclause 4.2.3.4:

"The bridging manager shall operate in a single threaded fashion (i.e., not issue more than one request at a time to the remote SMC device server). Queued requests received via the DT device primary port shall be queued in the local SMC device server and issued to the bridging manager one at a time. Moreover, if processing a single request by the local SMC device server requires issuing multiple requests to the remote SMC device server, then those requests shall be issued one at a time to the bridging manager."

The question I received was:

"What is meant by a request? A complete CDB-all the way through the status phase? Or is a request broken up in the different phases, Get SCSI CDB, Get SCSI Data, Send SCSI Data, and Send SCSI Status? Is a Task Management Function a request?"

The question came from someone who has apparently read SAM-3 and understands that a request is "a transaction invoking a service". I believe what we meant in the paragraph in question was "command" in several places in the paragraph. See section 5 for the proposed changes.

3.3 Cached information while bridging is disabled

An engineer from a different library manufacturer asked me if there were any rules about cached information while bridging is disabled. There were several questions including:

1. Must I respond to commands to the local SMC device server when bridging is disabled?
2. Must I send NOTIFY DATA TRANSFER DEVICE commands to the ADC device server on ready state changes even when bridging is disabled?
3. Can I make any assumptions about the state of the cached information in the local SMC device server when I enable bridging?

I did a little research and could find no rules in the standard that govern these conditions.

I don't believe we should place any additional requirements on either side with respect to question number 1. I believe that SPC-4 covers this well enough. An automation device that does not want to communicate with a bridging manager when bridging is disabled should implement the minimum command set required of all device servers spelled out in subclause 5.2 of SPC-4, INQUIRY, REPORT LUNS, and TEST UNIT READY. Each of these commands may return status or data that indicates there are no logical units available.

I believe questions 2 and 3 should be addressed in the standard. See section 6 for the proposed change.

3.4 Clarification on support for the NRSC, IDC, and MDC bits.

When bridging is enabled and caching is being used, the local SMC device server has the option of bridging all or some of the ready state, INQUIRY data, or MODE data. No method was provided for the automation to know how much of this data is cached. As such, the standard requires the automation application client to notify the ADC device server when any of this data changes (see 4.2.3.5): "If caching is enabled, the automation application client shall send the NOTIFY DATA TRANSFER DEVICE command (see 5.2) to the ADC device server when events occur that may change data cached by the local SMC device server."

Since the automation application client does not know if the data that changed is cached by the local SMC device server, it must report all changes. A good citizen would simply ignore any notifications that indicate data changed that it is not caching, but the standard is not clear in the regard. Section 7 proposes changes that make it mandatory to allow notifications of any possible cached data changes when caching is enabled.

4 Changes to fix the reservation problem in bridging

Add the following paragraph to subclause 4.2.3.1:

When ADI bridging is enabled, commands used to manage reservations are not passed through to the remote SMC device server because there is no method available to pass the true I_T nexus with they are associated. For this reason, reservations for the SMC device server are managed exclusively in the local SMC device server (see 4.2.3.2). An automation application client shall not enable ADI bridging in a DT device for a remote SMC device server that is accessible by an automation device primary port. An automation application client shall not enable ADI bridging in more than one DT device for a single remote SMC device server.

A bridging manager should enforce these requirements by establishing a reservation or persistent reservation with the remote SMC device server when ADI bridging is enabled. If a bridging manager attempts to establish a reservation or persistent reservation with a remote SMC device server and is unsuccessful, it shall return RESERVATION CONFLICT status to all commands from the local SMC device server.

5 Changes to address “requests” versus “commands”

In subclause 4.2.3.1, make the following changes:

> The local SMC device server receives a command or task management request via a DT device primary port. In processing the command or task management request, the local SMC device server may require the automation device to perform tasks. To do this, the local SMC device server passes requests to an application client in the DT device (i.e., the bridging manager). This communication is performed by means outside the scope of this standard. Using the ADT ports on the DT device and automation device, the bridging manager then invokes commands or task management requests on the remote SMC device server that resides in the automation device.
>
> The effect is that some or all commands and task management requests addressed to the local SMC device server are passed to the remote SMC device server through the ADT port. This may be used in low-cost automation devices that do not have automation device primary ports.

In subclause 4.2.3.2, make the following change:

> The local SMC device server shall support commands as required by the SCSI ~~Medium~~ Media Changer device type. Because the transport protocol connecting the bridging manager and the remote SMC device server may not carry information about which initiator port originated a command or task management request, the remote SMC device server is not able to implement the complete set of commands. Thus, the local SMC device server shall service commands and task management functions that require knowledge of the originating initiator port.

In subclause 4.2.3.4, make the following change:

> The bridging manager shall operate in a single threaded fashion (i.e., not issue more than one ~~request~~ command at a time to the remote SMC device server). ~~Queued requests~~ Commands received via the DT device primary port shall be queued in the local SMC device server and issued to the bridging manager one at a time. Moreover, if processing a single ~~request~~ command by the local SMC device server requires issuing multiple ~~requests~~ commands to the remote SMC

device server, then those ~~requests~~ commands shall be issued one at a time to the bridging manager.

In subclause 4.2.4.1, make the following change:

Load state (c) represents detection and acknowledgement by the DT device of medium presence, and that the DT device may now assume control of the medium and that the automation device should relinquish control of robotic access (e.g., this state may be reflected after medium movement caused by the automation device). An additional external stimulus is required to leave load state (c) (e.g., a ~~load request~~ LOAD UNLOAD command from the automation device).

In subclause 6.2.2.4.2, make the following change:

A SCSI unload hold override (SUHO) bit set to one indicates the HOLD bit in the SCSI LOAD UNLOAD command (see SSC-2) shall be ignored by the RMC device server and the medium shall not be ejected. A SUHO bit set to zero indicates the HOLD bit in the SCSI LOAD UNLOAD command shall control if the medium is ejected or not, as processed by the RMC device server. The SUHO bit shall not affect ~~unload requests~~ LOAD UNLOAD commands processed by the ADC device server.

6 Changes to define cached information with bridging disabled

Add the following test to the first paragraph in subclause 4.2.3.5:

The local SMC device server may preserve some data or status received from the remote SMC device server in a cache, in order to respond to certain commands without need for the bridging manager to invoke a command on the remote SMC device server (e.g., the local SMC device server may save the standard INQUIRY data from the remote SMC device server and return the data to any initiator port that requests it). The local SMC device server shall clear any data or status previously saved in a cache and shall not preserve any data or status in a cache while the ENABLE bit in the SMC Logical Unit descriptor (see 6.2.2.4.3) is set to zero.

7 Change to allow cached data changed notification

Add the following sentence to the second paragraph of subclause 4.2.3.5:

Caching of SMC ready state, standard INQUIRY data, VPD, and mode data is controlled by the CACHE bit in the SMC Logical Unit descriptor (see 6.2.2.4.3). When the CACHE bit is set to one, caching is enabled. If caching is enabled, the automation application client shall send the NOTIFY DATA TRANSFER DEVICE command (see 5.2) to the ADC device server when events occur that may change data cached by the local SMC device server. When the local SMC device server detects a possible change in the cached data, the local SMC device server shall discontinue using the cached data until the cached data has been updated. The local SMC device server shall issue any commands required to update the cache to the bridging manager before issuing any commands that the local SMC device server may have received from a DT device primary port and queued. An ADC device server that supports setting the CACHE bit to one in the SMC Logical Unit descriptor shall also support a NOTIFY DATA TRANSFER command with a value of one in any combination of the MDC, IDC, and NRSC bits.

Add the following sentences to the paragraphs that describe the NRSC, IDC, and MDC bits in subclause 5.2:

A mode data changed (MDC) bit set to one indicates that the contents of a mode page or mode parameter header reported by the remote SMC device server have changed. Upon receipt of this notification, the use of any cached mode data by the local SMC device server (see 4.2.3.2) shall be discontinued until the cached mode data has been refreshed. A MDC bit set to zero indicates that the contents have not changed. If the CACHE bit in the SMC Logical Unit descriptor is set to one, the device server shall support the MDC bit set to one, but may ignore the bit if mode data is not cached.

An INQUIRY data changed (IDC) bit set to one indicates that the contents of the standard INQUIRY data or of any VPD page reported by the remote SMC device server have changed. Upon receipt of this notification, the use of any cached INQUIRY data or VPD pages shall be discontinued until the cached data or pages have been refreshed. An IDC bit set to zero indicates that the contents have not changed. If the CACHE bit in the SMC Logical Unit descriptor is set to one, the device server shall support the IDC bit set to one, but may ignore the bit if INQUIRY data is not cached.

A not ready state changed (NRSC) bit set to one indicates that the remote SMC device server has entered the not accessible state, per the description of caching SMC data and status (see 4.2.3.5). A NRSC bit set to one may also indicate that the remote SMC device server was already in the not accessible state and the sense data changed. When the NRSC bit is set to one, the ASC and ASCQ fields shall contain additional sense data appropriate to the condition. Upon receipt of this notification, the cached ready state and additional sense data shall be updated. An NRSC bit set to zero indicates that the remote SMC device server has not entered the not accessible state, nor has the additional sense data changed if already in the not accessible state. If the CACHE bit in the SMC Logical Unit descriptor is set to one, the device server shall support the NRSC bit set to one, but may ignore the bit if ready state is not cached.