To: INCITS Technical Committee T10
From: Fred Knight, Network Appliance
Email: knight@netapp.com
Date: May 7, 2008
Subject: SPC – Persistent Reservations SPEC_I_PT clarification

1. ***Revision history***
   Revision 0 (May 7, 2008) First revision

2. ***Related documents***
   spc4r14 – SCSI Primary Commands – 4

3. ***Overview***
   During the discussions of the Team Reservation proposal from IBM at the Raleigh CAP meeting, it became clear that some portions of SPEC_I_PT were not clearly defined.

   Specifically, the error cases associated with the inability to register some of the initiators in the SPEC_I_PT list. The specific text is:

   "If a registration fails for any initiator port (e.g., if the logical unit does not have enough resources available to hold the registration information), none of the other registrations shall be made."

   The lack of clarity has to do with which other registrations the phrase "none of the other registrations" is referring to. Does it refer to:
   a. Only the other registrations in the SPEC_I_PT list, or
   b. Only the remaining registrations in the list, or
   c. Any other registration that might have occurred as a result of this PR_OUT command?

   I believe case b above is clear not the correct interpretation.

   The error case immediately following table 162 describes a failure case where it would be possible to do some registrations, but not others. It is clear in this text that the error checking occurs BEFORE any possible registration is made, and as a result, it is clear that no registration what so ever will be made. However, this text specifically includes the error code that should be returned. The earlier text does not.

   An example follows:

PR_OUT – REGISTER:

I_T that sends command	SPEC_I_PT #1	SPEC_I_PT #2	SPEC_I_PT #3	SPEC_I_PT #4

Case 1 – The initiator sends the REGISTER command with SPEC_I_PT set to one and includes 4 transport IDs. If the target has resources to store 3 registrations, and "the other registrations" applies to only those in the SPEC_I_PT list, then it tries to save:

1) the I_T that sends the command
2) SPEC_I_PT #1
3) SPEC_I_PT #2

BUT, since it failed for "any initiator port" then, none of "the other" registrations happen, so… the registration for SPEC_I_PT #1 goes away and the registration for SPEC_I_PT #2 goes away, but the I_T that sends the command is retained. So, we have in fact made 1 registration.

Does the lack of a specified error code indicate this is the preferred interpretation? However, since there is no indication that some initiators were not registered, this means any initiator using this feature would be registered, but to find out if the initiators in the SPEC_I_PT list were registered, the initiator must issue a PR_IN command after the SPEC_I_PT is sent to find out what was registered.

This seems inconsistent that the operation would partially succeed (the issuing initiator would be registered) and partially fail (the SPEC_I_PT list would not be registered), and there would be no indication that this occurred.

Case 2 – The initiator sends the REGISTER command with SPEC_I_PT set to one and includes 4 transport IDs. If the target has resources to store 3 registrations, and "the other registrations" applies to all other, then it tries to save:

1) the I_T that sends the command
2) SPEC_I_PT #1
3) SPEC_I_PT #2

BUT, since it failed for "any initiator port" then, none of "the other" registrations happen, so… the registration for all of the above goes away – even the I_T that sends the command. None of any other registrations has been made.

Does the other failure case (where no registrations occur at all) indicate that this is the preferred interpretation? If this is correct, then a failure code must be defined for the currently unspecified case.

In fact, the model clause 5.6.7 Registering, contains the following text:

"If a PERSISTENT RESERVE OUT command with a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action is attempted, but there are insufficient device server resources to complete the operation, then the command shall be terminated with CHECK

CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT REGISTRATION RESOURCES."

Table 45 in this section clearly shows that this text applies to cases where the SPEC_I_PT bit is set to one. This text makes is quite clear that if sufficient resources do not exist, the entire command must fail and no registrations what so ever will occur.

This proposal attempts to clarify this situation by duplicating the specified error code from the model clause 5.6.7 into the text in 6.14.3.

Proposal:

6.14.3 Basic PERSISTENT RESERVE OUT parameter list

<…>

If the Specify Initiator Ports (SPEC_I_PT) bit is set to zero, the device server shall apply the registration only to the I_T nexus that sent the PERSISTENT RESERVE OUT command. If the SPEC_I_PT bit is set to one for any service action except the REGISTER service action, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. If the SPEC_I_PT bit is set to one for the REGISTER service action, the additional parameter data (see table 162) shall include a list of transport IDs and the device server shall also apply the registration to the I_T nexus for each initiator port specified by a TransportID. If a registration fails for any initiator port (e.g., if the logical unit does not have enough resources available to hold the registration information), no~~ne of the other~~ registrations shall be made, and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST and the additional sense code set to INSUFFICIENT REGISTRATION RESOURCES.

Table 162 — PERSISTENT RESERVE OUT specify initiator ports additional parameter data

Bit / Byte	7	6	5	4	3	2	1	0
24	TRANSPORTID PARAMETER DATA LENGTH (n - 27)							
27								
	TransportIDs list							
28	TransportID [first]							
	⋮							
	TransportID [last]							
n								

The TRANSPORTID PARAMETER DATA LENGTH field specifies the number of bytes of TransportIDs that follow.

The command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST:

a) If the value in the PARAMETER LIST LENGTH field in the CDB does not include all of the additional parameter list bytes specified by the TRANSPORTID PARAMETER DATA LENGTH field; or
b) If the value in the TRANSPORTID PARAMETER DATA LENGTH field results in the truncation of a TransportID.

The format of a TransportID is specified in 7.5.4.

<…>