To: T10 Technical Committee
From: Rob Elliott, HP (elliott@hp.com)
Date: 10 March 2004
Subject: T10/03-342r5 SPC-3 Persistent reservations read full status

Revision History

Revision 0 (3 October 2003) first revision

Revision 1 (10 October 2003) add that PR IN/READ RESERVATION returns a key of 0 when an all registrants type reservation is present. Require the scope, type, and scope-specific address fields be set to zero in the full status descriptor when there is no reservation active.

Revision 2 (12 December 2003) incorporated comments from November CAP WG - assume that element reservations are obsolete (per 03-167).

Revision 3 (30 December 2003) updated to match 03-344r2's proposed change to the relative port field size (2 bytes rather than 4). Changed to use 0000h value rather than FFFFh to mean all target ports (0h is reserved as a relative port number since they start at 1h; FFFFh is a valid number)

Revision 4 (22 February 2004) updated baseline text to spc3r17 (which got rid of element reservations).

Revision 5 (10 March 2004) incorporated comments from March CAP WG – added an ALL TARGET PORTS bit rather than overload the value of 0000h in the RELATIVE TARGET PORT IDENTIFIER field.

Related Documents

spc3r17 – SCSI Primary Commands – 3 revision 17
03-344 - SPC-3 SAM-3 Report all initiator and target ports

Overview

PERSISTENT RESERVE IN defines two service actions to retrieve status about the current reservations in the logical unit:

- READ KEYS - returns all keys that are registered (just the keys)
- READ RESERVATION - returns the key of the reservation holder(s) and the type and scope of the reservation (assuming 03-167 is adopted)

However, what is really registered or the reservation holder is an I_T nexus, not just a key. Neither of these service actions identify the I_T nexus holding the key (or reservation).

Proposal

Add a PERSISTENT RESERVE IN service action that returns a list of all the I_T nexuses that are registered and identifies which ones are reservation holders (there may be more than one for all-registrants types).

Initiator ports are described with Transport IDs; target ports are described with relative target port identifiers.

An application can convert the relative target port identifier into a target port identifier by other means (see 03-344).

In case the logical unit is taking advantage of the All Target Ports feature and is not differentiating the Ts in the I_T nexus, it can return a single descriptor that covers one initiator port and all target ports.

Suggested Changes

5.6 Reservations

[5.6.1 Persistent Reservations overview]

[5.6.2 Exceptions to SPC-2 RESERVE and RELEASE behavior]

[5.6.3 Preserving persistent reservations and registrations]

5.6.4 Finding persistent reservations and reservation keys

5.6.4.1 Summary of commands for finding persistent reservations and reservation keys

The application client may obtain information about the persistent reservation and the reservation keys (i.e., registrations) that are present within a device server by issuing a PERSISTENT RESERVE IN command with a READ RESERVATION service action ~~or~~, a READ KEYS service action, or a READ FULL STATUS service action.

5.6.4.2 Reporting reservation keys

An application client may issue a PERSISTENT RESERVE IN command with READ KEYS service action to determine if any I_T nexuses have been registered with a logical unit through any target port.

In response to a PERSISTENT RESERVE IN with READ KEYS service action the device server shall report the following:

a) The current PRgeneration value (see 6.11.3); and

b) The reservation key for every I_T nexus that is currently registered regardless of the target port through which the registration occurred.

The PRgeneration value allows the application client to verify that the configuration of the I_T nexuses registered with a logical unit has not been modified.

~~The application client may examine the reservation keys to identify relationships between I_T nexuses based on mechanisms that are outside the scope of this standard.~~ Duplicate reservation keys shall be reported if multiple I_T nexuses are registered using the same reservation key.

If an application client uses a different reservation key for each I_T nexus, the application client may use the reservation key to uniquely identify an I_T nexus.

[Editor's note: this proposal adds a mechanism to the standard, so the "outside the scope" line is no longer true. The added paragraph is from 5.6.2.3.3 (slightly modified).]

5.6.4.3 Reporting persistent reservations

An application client may issue a PERSISTENT RESERVE IN command with READ RESERVATION service action to receive the persistent reservation information.

In response to a PERSISTENT RESERVE IN command with READ RESERVATION service action the device server shall report the following ~~as an uninterrupted series of actions~~ information for the persistent reservation, if any:

[Editor's note: the "uninterrupted" warning is usually for setting things, not reading.]

a) The current PRgeneration value (see 6.11.3);

b) The registered reservation key, if any, associated with the I_T nexus that holds the persistent reservation. If the persistent reservation is an all registrants type, the registered reservation key reported shall be zero;

c) The scope and type of the persistent reservation, if any.

If an application client uses a different reservation key for each I_T_L nexus the application client may use the reservation key to associate the persistent reservation with the initiator port that holds the persistent reservation. This association is done using techniques that are outside the scope of this standard.

5.6.4.4 Reporting full status [new section]

An application client may issue a PERSISTENT RESERVE IN command with READ FULL STATUS service action to receive all information about registrations and the persistent reservation, if any.

In response to a PERSISTENT RESERVE IN with READ FULL STATUS service action the device server shall report the current PRgeneration value (see 6.11.3) and, for every I_T nexus that is currently registered, the following information:

a) The registered reservation key;

b) Whether the I_T nexus is a persistent reservation holder;

c) The scope and type of the persistent reservation, if the I_T nexus is a persistent reservation holder;

e) The relative target port identifier identifying the target port of the I_T nexus; and

f) A TransportID identifying the initiator port of the I_T nexus.

[5.6.5 Registering]

[5.6.6 Reserving]

[5.6.7 Persistent reservation holder]

[**5.6.8 Releasing persistent reservations and removing registrations**]

6.11 PERSISTENT RESERVE IN command

6.11.1 PERSISTENT RESERVE IN command introduction

The PERSISTENT RESERVE IN command (see table 96) is used to obtain information about persistent reservations and reservation keys (i.e., registrations) that are active within a device server. This command is used in conjunction with the PERSISTENT RESERVE OUT command (see 6.12).

Table 94 - PERSISTENT RESERVE IN command

[table not shown]

The PERSISTENT RESERVE IN parameter data includes a field that indicates the number of parameter data bytes available to be returned. The ALLOCATION LENGTH field in the CDB indicates how much space has been allocated for the returned parameter list. An allocation length that is not sufficient to contain the entire parameter list shall not be considered an error. If the complete list is required, the application client should send a new PERSISTENT RESERVE IN command with allocation length large enough to contain the entire list.

6.11.2 PERSISTENT RESERVE IN service actions

6.11.2.1 Summary of PERSISTENT RESERVE IN service actions

The service action codes for the PERSISTENT RESERVE IN command are defined in table 97.

Table 97 — PERSISTENT RESERVE IN service action codes

Code	Name	Description	References
00h	READ KEYS	Reads all registered reservation keys (i.e., registrations) as described in 5.6.2.3.2	6.11.2.2 6.11.3
01h	READ RESERVATION	Reads the current persistent reservation as described in 5.6.2.3.3	6.11.2.3 6.11.4
02h	REPORT CAPABILITIES	Returns capability information	6.11.2.4 6.11.5
03h	READ FULL STATUS	Reads complete information about all registrations and persistent reservations	6.11.2.x
~~03h~~ 04h - 1Fh	Reserved	Reserved	

[Editor's note: the structure here seems odd. There are 3 short sections (6.11.2.x) providing one paragraph overviews of each service action, then 3 long sections (6.11.x) describing the parameter data for each service action in detail. I suggest restructuring it like the READ COPY

RESULTS command to combine the overview and parameter data text for each service action and remove one layer of hierarchy.

The new structure would be:

6.11 PR IN

6.11.1 PR IN command introduction [contents of 6.11.1 and 6.11.2.1]

6.11.2 READ KEYS service action [contents of 6.11.2.2 and 6.11.3

6.11.3 READ RESERVATION service action [contents of 6.11.2.3 and 6.11.4]

6.11.3 REPORT CAPABILITIES service action [contents of 6.11.2.4 and 6.11.5]

6.11.3 READ FULL STATUS service action [contents of 6.11.2.x and 6.11.x]

]

6.11.2.2 READ KEYS

The READ KEYS service action requests that the device server return a parameter list containing a header and a list of each currently registered I_T nexus' reservation key. If multiple I_T nexuses have registered with the same key, then that key value shall be listed multiple times, once for each such registration.

For more information on READ KEYS see 5.6.2.3.2.

6.11.2.3 READ RESERVATION

The READ RESERVATION service action requests that the device server return a parameter list containing a header and the persistent reservation, if any, that is present in the device server.

For more information on READ RESERVATION see 5.6.2.3.3.

6.11.2.4 REPORT CAPABILITIES

The REPORT CAPABILITIES service action requests that the device server return information on persistent reservation features.

6.11.2.4x READ FULL STATUS [new section]

The READ FULL STATUS service action requests that the device server return a parameter list describing the registration and persistent reservation status of each currently registered I_T nexus for the logical unit.

6.11.3 PERSISTENT RESERVE IN parameter data for READ KEYS
[no changes]

6.11.4 PERSISTENT RESERVE IN parameter data for READ RESERVATION
[no changes]

6.11.5 PERSISTENT RESERVE IN parameter data for REPORT CAPABILITIES

[no changes]

6.11.6 PERSISTENT RESERVE IN parameter data for READ FULL STATUS [new section]

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the READ FULL STATUS service action is shown in table xx.

Table xx — PERSISTENT RESERVE IN parameter data for READ FULL STATUS

	7	6	5	4	3	2	1	0
0	(MSB)							
3				PRGENERATION				(LSB)
4	(MSB)							
7				ADDITIONAL LENGTH (n - 7)				(LSB)
8								
n				Full status descriptor(s)				

The PRGENERATION field shall be as defined for the PERSISTENT RESERVE IN READ KEYS parameter data (see 6.11.3).

The ADDITIONAL LENGTH field contains a count of the number of bytes to follow in full status descriptor(s). If the allocation length specified by the PERSISTENT RESERVE IN command is not sufficient to contain the entire parameter list, then only the first portion of the list (byte 0 to the allocation length) shall be sent to the application client. The incremental remaining bytes shall be truncated, although the ADDITIONAL LENGTH field shall still contain the actual number of bytes of full status descriptor(s) and shall not be affected by the truncation. This shall not be considered an error.

The format of the full status descriptors is defined in table yy. The device server shall return a full status descriptor covering each registered I_T nexus. Each full status descriptor describes one or more registered I_T nexuses.

Table yy — PERSISTENT RESERVE IN full status descriptor

	7	6	5	4	3	2	1	0
0	(MSB)							
7				RESERVATION KEY				(LSB)
8								
11				Reserved				
12				Reserved				HOLDS RESERVATION
13		SCOPE			TYPE			
14								
17				Reserved				
18	(MSB)							
19				RELATIVE TARGET PORT				(LSB)
20	(MSB)							
23				ADDITIONAL DESCRIPTOR LENGTH (n - 23)				(LSB)
24								
n				TRANSPORTID				

[Editor's notes: bytes 0-15 are laid out like bytes 8-23 of the READ RESERVATION data, with the HOLDS RESERVATION bit added and obsolete fields marked reserved]

The RESERVATION KEY field contains the reservation key.

The SCOPE field and TYPE field are as defined in the READ RESERVATION parameter data (see 6.11.4) if this I_T nexus is a reservation holder (i.e., if the HOLDS RESERVATION bit is set to one) and shall be set to zero and ignored by the application client if this I_T nexus is not a reservation holder (i.e., if the HOLDS RESERVATION bit is set to zero).

An ALL TARGET PORTS bit set to zero indicates that this descriptor represents a single I_T nexus. An ALL TARGET PORTS bit set to one indicates that:

a) This descriptor represents all the I_T nexuses involving:

A) the initiator port specified by the TRANSPORTID field; and

B) each target port in the SCSI target device;

b) The I_T nexuses are registered with the same reservation key;

c) The I_T nexuses are either all reservation holders or all not reservation holders; and

c) The RELATIVE TARGET PORT field shall be ignored by the application client.

The device server is not required to return an ALL TARGET PORTS bit set to one when it is allowed to do so; it may return individual descriptors for each I_T nexus.

A HOLDS RESERVATION bit of one indicates that all I_T nexuses described by this full status descriptor are registered and are persistent reservation holders. A HOLDS RESERVATION bit of zero indicates that all I_T nexuses described by this full status descriptor are registered but are not persistent reservation holders.

The RELATIVE TARGET PORT field contains the relative port identifier of the target port that is part of the I_T nexus, if the ALL TARGET PORTS bit is set to one.

The ADDITIONAL DESCRIPTOR LENGTH field contains a count of the number of bytes to follow in the descriptor (i.e., the size of the TransportID).

The TRANSPORTID field contains a TransportID (see 7.5.4) identifying the initiator port that is part of the I_T nexus or I_T nexuses described by this descriptor.