Date: July 14, 2004

To: T10 Committee (SCSI)

From: George Penokie (IBM/Tivoli)

Subject: SPC-3: Third-party Persistent Reservations

1 Overview

1.1 Revision History

Revision 0 (3 October 2003) first revision

Revision 1 (16 October 2003) incorporated feedback from 10/14 conference call. Changed from adding a GIVE bit to REGISTER to a new service action called REGISTER AND MOVE. Changed from adding a TAKE bit to REGISTER to adding Specify Initiator Ports support to PREEMPT.

Revision 2 (31 October 2003) incorporated feedback from 10/28 conference call.

Revision 3 (11 December 2003) incorporated feedback from November CAP WG. Prohibit unregister for a specified I_T nexus other than the one used for the PR OUT command.

Revision 4 (4 March 2004) updated to base of spc3r17, which dropped element reservations and raised the model section numbers one level.

Revision 5 (May 2004) major restructuring of the proposal that only defines a REGISTER AND MOVE and defines a new parameter list for the REGISTER AND MOVE that defines the I_T nexus that the reservation is moved to.

Revision 6 (May 2004) incorporated comments from the May working group.

Revision 7 (July 2004) incorporated comments from the July working group.

1.2 Related Documents

spc3r17 SCSI Primary Commands – 3 revision 17
03-231 Two Persistent Reservations problems - latest information (Roger Cummings, Veritas)
03-233 New PR Out Service Action Proposal (Roger Cummings, Veritas)
03-321 Persistent Reservations Proposals (Roger Cummings, Veritas)
03-342 SPC-3 Persistent reservations read full status (Rob Elliott, HP)
03-344 SPC-3 Report all target port identifiers (Rob Elliott, HP)
03-353 SPC-3 Report initiator port identifiers (Rob Elliott, HP)
03-354 SPC-3 Specific initiator ports in EXTENDED COPY target descriptors (Rob Elliott, HP)

1.3 Overview

Classic RESERVE/RELEASE reservations included a third-party reservation feature that let one initiator port hand over its reservation to another initiator port. This is useful for third party copy managers (EXTENDED COPY). However, this feature is not available in persistent reservations.

Figure 1 — Third-party copy

03-233 proposed a MOVE service action, specifying the reservation key to move. However, this isn't precise enough if more than one I_T is registered with the same reservation key. The Specify Initiator ports feature can be used to fix this.

Also, it requires the copy manager to register ahead of time. There is an EXTENDED COPY descriptor that directs the recipient to register with a specified reservation key. However, this is an extra step that can be eliminated.

1.4 Proposal

Add a REGISTER and MOVE service actions to both register a specified I_T nexus and move the existing reservation to it, all in one step. ~~Add Specify Initiator Ports functionality to PREEMPT to take back the reservation from a specific I_T nexus~~.

Figure 2 — Third party persistent reservation

After it has become a reservation holder through an I_T, the backup application uses a PERSISTENT RESERVE OUT command REGISTER AND MOVE service action with:

a) parameter data specifying the copy manager's target port; ~~SPEC_I_PT bit set to 1~~
b) parameter data specifying the copy manager's initiator port;
c) SERVICE ACTION RESERVATION KEY set to a reservation key for the copy manager; and
d) RESERVATION KEY set to its own reservation key (normal rules for these service actions);

to move its current persistent reservation to the specified initiator port.

It is an error to use this service action if the initiator is not a reservation holder or if there is more than one reservation holder (i.e., all registrants) ~~if more than one initiator port is specified in the parameter list~~.

The copy manager becomes registered with the specified reservation key and becomes a reservation holder in place of ~~(or in addition to, depending on the reservation type)~~ the backup application. The application should use the same reservation key as the backup application, so if the reservation key is preempted, both the copy manager and the backup application are preempted together.

To move the reservation to the original holder of the reservation ~~When it wants to revoke the handover,~~ the backup application uses a ~~PERSISTENT RESERVE OUT command PREEMPT service action or PREEMPT AND ABORT service action with:~~ PERSISTENT RESERVE OUT command REGISTER AND MOVE service action with:

a) parameter data specifying the 3rd party originators target port ~~SPEC_I_PT bit set to 1~~
b) parameter data specifying the ~~copy manager's~~ originators initiator port
c) SERVICE ACTION RESERVATION KEY set to the copy manager's reservation key
d) RESERVATION KEY set to its own key (normal rules for these service actions)

to unregister the copy manager and make the backup application the reservation holder.

The service actions are subject to all their normal rules about which I_T nexus is allowed to be used to send them and what the RESERVATION KEY must be.

1.5 Suggested Changes

5.6 Reservations

5.6.1 Persistent Reservations overview

Reservations may be used to allow a device server to process commands from a selected set of I_T nexuses (i.e., combinations of initiator ports accessing target ports) and reject commands from I_T nexuses outside the selected set. The device server uniquely identifies I_T nexuses using protocol specific mechanisms.

Application clients may add or remove I_T nexuses from the selected set using reservation commands. If the application clients do not cooperate in the reservation protocol, data may be unexpectedly modified and deadlock conditions may occur.

The persistent reservations mechanism allows multiple application clients communicating through multiple I_T nexuses to protect reservation operations across SCSI initiator device failures, which usually involve logical unit resets and involve I_T nexus losses. Persistent reservations persist across recovery actions, to provide application clients with more detailed control over reservations recovery. Persistent reservations are not reset by hard reset, logical unit reset, or I_T nexus loss.

The persistent reservation held by a failing I_T nexus may be preempted by another I_T nexus as part of the recovery process. Persistent reservations shall be retained by the device server until released, preempted, or cleared by mechanisms specified in this standard. Optionally, persistent reservations may be retained when power to the target is removed.

The PERSISTENT RESERVE OUT and PERSISTENT RESERVE IN commands provide the basic mechanism for dynamic contention resolution in systems with multiple initiator ports using logical units with SCSI 3 multiple target ports.

Before a persistent reservation may be established, an application client shall register each I_T nexus with a device server using a reservation key. Reservation keys are necessary to allow:

a) Authentication of subsequent PERSISTENT RESERVE OUT commands;
b) Identification of other I_T nexuses that are registered;
c) Identification of the reservation key(s) that have an associated persistent reservation;
d) Preemption of a persistent reservation from a failing or uncooperative I_T nexus; and
e) Multiple I_T nexuses to participate in a persistent reservation.

The reservation key provides a method for the application client to associate a protocol-independent identifier with a registered I_T nexus. The reservation key is used in the PERSISTENT RESERVE IN command to identify which I_T nexuses are registered and which I_T nexus, if any, holds the persistent reservation. The reservation key is used in the PERSISTENT RESERVE OUT command to register an I_T nexus, to verify the I_T nexus being used for the PERSISTENT RESERVATION OUT command is registered, and to specify which registrations or persistent reservation to preempt.

Reservation key values may be used by application clients to identify registered I_T nexuses, using application specific methods that are outside the scope of this standard. This standard provides the ability to register no more than one reservation key per I_T nexus. Multiple I_T nexuses may use the same reservation key for a logical unit accessed through the same target port. Multiple initiator ports may use the same reservation key value for a logical unit accessed through the same target ports. An initiator port may use the same reservation key value for a logical unit accessed through different target ports. The logical unit shall maintain a separate reservation key for each I_T nexus, regardless of the reservation key's value.

An application client may register an I_T nexus with multiple logical units in a SCSI target device using any combination of unique or duplicate reservation keys. These rules provide the ability for an application client to preempt multiple I_T nexuses with a single PERSISTENT RESERVE OUT command, but they do not provide the ability for the application client to uniquely identify the I_T nexuses using the PERSISTENT RESERVE commands.

See table 105 in 6.12.2 for a list of PERSISTENT RESERVE OUT service actions. See table 97 in 6.11.2.1 for a list of PERSISTENT RESERVE IN service actions.

The scope of a persistent reservation shall be the entire logical unit.

Persistent reservations may be further qualified by restrictions on types of access (e.g., read, write). However, any restrictions based on the type of reservation are independent of the scope of the reservation.

Reservation restrictions are placed on commands as a result of access qualifiers associated with the type of reservation. The details of which commands are allowed under what types of reservations are described in table 1.

In table 1 and table 2 the following key words are used:

allowed: Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present should complete normally.

conflict: Commands received from I_T nexuses not holding the reservation or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present shall not be performed and the device server shall terminate the command with a RESERVATION CONFLICT status.

Commands from I_T nexuses holding a reservation should complete normally. The behavior of commands from registered I_T nexuses when a registrants only or all registrants type persistent reservation is present is specified in table 1 and table 2.

An unlinked command shall be checked for reservation conflicts before the task containing that command enters the enabled task state. The reservation state as it exists when the first command in a group of linked commands enters the enabled task state shall be used in checking for reservation conflicts for all the commands in the task. Once a task has entered the enabled task state, the command or commands comprising that task shall not be terminated with a RESERVATION CONFLICT due to a subsequent reservation. Any command in a group of linked commands that changes the reservation state shall be the last command in the group.

For each command, this standard or other command standard (see 3.1.18) defines the conditions that result in RESERVATION CONFLICT. Command standards define the conditions either in the device model (preferred) or in the descriptions each specific command.

Table 1 — SPC commands that are allowed in the presence of various reservations (part ? of ?)
Table not shown

Table 2 — PERSISTENT RESERVE OUT service actions that are allowed in the presence of various reservations

Command Service Action	Addressed LU has a persistent reservation held by another I_T nexus	
	Command is from a registered I_T nexus	Command is from a not registered I_T nexus
CLEAR	Allowed	Conflict
PREEMPT	Allowed	Conflict
PREEMPT AND ABORT	Allowed	Conflict
REGISTER	Allowed	Allowed
REGISTER AND IGNORE EXISTING KEY	Allowed	Allowed
REGISTER AND MOVE	Allowed	Conflict
RELEASE	Allowed [a]	Conflict
RESERVE	Conflict	Conflict
Key: **LU**=Logical Unit		
[a] The reservation is not released (see 5.6.10.2).		

The time at which a reservation is established 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 established. A reservation may apply to some or all of the tasks in the task set before the completion of the reservation command. The reservation shall apply to all tasks received by the device server after successful completion of the reservation command. Any persistent reserve service action shall be performed as a single indivisible event.

Multiple persistent reserve service actions may be present in the task set at the same time. The order of processing of such service actions is defined by the task set management requirements defined in SAM-3, but each is processed as a single indivisible command without any interleaving of actions that may be required by other reservation commands.

5.6.2 Third party persistent reservations

Except for all registrants type reservations, a reservation holder (see 5.6.7) may move a persistent reservation to a third party (e.g., a copy manager supporting the EXTENDED COPY command) using the REGISTER AND MOVE service action.

~~To accomplish a third party persistent reservation the following steps are recommended:~~

~~1) Backup application uses the REGISTER service action to register an I_T nexus with a logical unit (e.g., a tape drive logical unit);~~
~~2) Backup application uses the RESERVE service action to establish a persistent reservation with the Exclusive Access type;~~
~~3) Backup application prepares the logical unit for access (e.g., ensures the tape is loaded and rewound);~~
~~4) Backup application uses the REGISTER AND MOVE service action to register the I_T nexus that the copy manager is expected to use and moves the persistent reservation to that I_T nexus;~~
~~5) Backup application sends the EXTENDED COPY command to the copy manager that includes an I_T nexus that the copy manager is expected to use in a REGISTER AND MOVE service action to return the persistent reservation to the original I_T nexus;~~
~~6) Copy manager accesses the logical unit (e.g., writes to the tape); and~~
~~7) The copy manager issues a REGISTER AND MOVE service action, using the reservation key and I_T nexus received by the copy manager, to move the persistent reservation back to the original I_T nexus.~~

5.6.3 Exceptions to SPC-2 RESERVE and RELEASE behavior

This subclause defines exceptions to the behavior of the RESERVE and RELEASE commands defined in SPC-2. The RESERVE and RELEASE commands are obsolete in this standard, except for the behavior defined in this subclause. Device servers that operate using the exceptions described in this subclause shall set the CRH bit to one in the parameter data returned by the REPORT CAPABILITIES service action of the PERSISTENT RESERVE IN command (see 6.11.4).

A RELEASE(6) or RELEASE(10) command shall complete with GOOD status, but the persistent reservation shall not be released, if the command is received from:

a) An I_T nexus that is a persistent reservation holder (see 5.6.9); or
b) An I_T nexus that is registered if a registrants only or all registrants type persistent reservation is present.

A RESERVE(6) or RESERVE(10) command shall complete with GOOD status, but no reservation shall be established and the persistent reservation shall not be changed, if the command is received from:

a) An I_T nexus that is a persistent reservation holder; or
b) An I_T nexus that is registered if a registrants only or all registrants type persistent reservation is present.

In all other cases, a RESERVE(6) command, RESERVE(10) command, RELEASE(6) command, or RELEASE(10) command shall be processed as defined in SPC-2.

5.6.4 Preserving persistent reservations and registrations

The application client may request activation of the persist through power loss device server capability to preserve the persistent reservation and registrations across power cycles by setting the APTPL bit to one in the PERSISTENT RESERVE OUT parameter data sent with a REGISTER service action, ~~or~~ REGISTER AND IGNORE EXISTING KEY service action, or REGISTER AND MOVE service action.

After the application client enables the persist through power loss capability the device server shall preserve the persistent reservation, if any, and all current and future registrations associated with the logical unit to which the REGISTER service action, ~~or~~ REGISTER AND IGNORE EXISTING KEY service action, or REGISTER AND MOVE service action was addressed until an application client disables the persist through power loss capability. The APTPL value from the most recent successfully completed REGISTER service action, ~~or~~ REGISTER AND

IGNORE EXISTING KEY service action, or REGISTER AND MOVE service action from any application client shall determine the logical unit's behavior in the event of a power loss.

The device server shall preserve the following information for each existing registration across any hard reset, logical unit reset, or I_T nexus loss, and if the persist through power loss capability is enabled, across any power cycle:

a) For SCSI transport protocols where initiator port names (see 3.1.49) are required, the initiator port name; otherwise, the initiator port identifier (see 3.1.48);
b) Reservation key; and
c) Indication of the target port to which the registration was applied.

The device server shall preserve the following information about the existing persistent reservation across any hard reset, logical unit reset, or I_T nexus loss, and if the persist through power loss capability is enabled, across any power cycle:

a) For SCSI transport protocols where initiator port names are required, the initiator port name; otherwise, the initiator port identifier;
b) Reservation key;
c) Scope;
d) Type; and
e) Indication of the target port through which the reservation was established.

NOTE 1 - The scope of a persistent reservation is always LU_SCOPE. For an all registrants type persistent reservation, only the scope and type need to be preserved.

The capability of preserving persistent reservations and registrations across power cycles requires the use of a nonvolatile memory within the SCSI device. Any SCSI device and logical unit that supports the persist through power loss capability of persistent reservation and has nonvolatile memory that is not ready shall allow the following commands into the task set:

a) INQUIRY;
b) LOG SENSE;
c) READ BUFFER;
d) REPORT LUNS;
e) REQUEST SENSE;
f) START STOP UNIT (with the START bit set to one and POWER CONDITIONS field value of 0h); and
g) WRITE BUFFER.

When nonvolatile memory has not become ready since a power cycle, commands other than those listed in this subclause shall return CHECK CONDITION status. The sense key shall be set to NOT READY and the additional sense code shall be set as described in table 172 (see 6.31).

5.6.5 Finding persistent reservations and reservation keys

5.6.5.1 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.

5.6.5.2 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 6.11.2); 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.

5.6.5.3 Reporting the persistent reservation

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:

a) The current PRgeneration value (see 6.11.2);
b) The registered reservation key, if any, associated with the I_T nexus that holds the persistent reservation; and
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.

5.6.6 Registering

To establish a persistent reservation the application client shall first register an I_T nexus with a logical unit. An application client registers with a logical unit by issuing a PERSISTENT RESERVE OUT command with REGISTER service action or REGISTER AND IGNORE EXISTING KEY service action, ~~or REGISTER AND MOVE service action.~~

If the I_T nexus has not yet established a reservation key or the reservation key and registration have been removed, the registration of on one or more I_T nexus is accomplished by issuing a PERSISTENT RESERVE OUT command with REGISTER service action as defined in table 3. ~~with the following parameters:~~

~~a) APTPL bit optionally set to one;~~
~~b) RESERVATION KEY field set to zero; and~~
~~c) SERVICE ACTION RESERVATION KEY field set to a non-zero value.~~

If the I_T nexus has an established registration ~~it~~ an application client may change ~~its~~ the reservation key on one or more I_T nexus. This is accomplished by issuing a PERSISTENT RESERVE OUT command with REGISTER service action as defined in table 4. ~~with the following parameters:~~

~~a) APTPL bit optionally set to one;~~
~~b) RESERVATION KEY field set to the value of the reservation key that is registered for the I_T_L nexus; and~~
~~c) SERVICE ACTION RESERVATION KEY field set to a non-zero value.~~

~~If the SERVICE ACTION RESERVATION KEY field is set to zero, the registration shall be removed (see 5.6.10.1.1).~~

Alternatively, an application client may establish a reservation key for ~~an~~ one or more I_T nexus without regard for whether one has previously been established by issuing a PERSISTENT RESERVE OUT command with REGISTER AND IGNORE EXISTING KEY service action as defined in table 3 and table 4. ~~and the following parameters:~~

~~a) APTPL bit optionally set to one; and~~
~~b) SERVICE ACTION RESERVATION KEY field set to a non-zero value.~~

~~If the SERVICE ACTION RESERVATION KEY field is set to zero and the I_T_L nexus is registered, the registration shall be removed (see 5.6.10.1.1).~~

If the I_T nexus has an established registration an application client may remove the reservation key (see 5.6.10.1.1). This is accomplished by issuing a PERSISTENT RESERVE OUT command with a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action as defined in table 4.

~~If the I_T nexus has an established registration an application client may copy that established registration to another I_T nexus by issuing a PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action as defined in table 4.~~

Table 3 — Register behaviors for a service action received on an unregistered I_T nexus

Service action [a] [b] [c]	RESERVATION KEY field	SERVICE ACTION RESERVATION KEY field	Result
REGISTER	zero	zero	Return GOOD status
		non-zero	Register all the specified I_T nexus with the value specified in the SERVICE ACTION RESERVATION KEY field.
	non-zero	ignore	Return RESERVATION CONFLICT status
REGISTER AND IGNORE EXISTING KEY	any	zero	Return GOOD status
		non-zero	Register all the specified I_T nexus with the value specified in the SERVICE ACTION RESERVATION KEY field.
~~REGISTER AND MOVE~~	~~any~~	~~ignore~~	~~Return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.~~

[a] APTPL bit may be set to one.
[b] For usage of the SPEC_I_PT bit see 6.12.3.
[c] For usage of the ALL_TG_PT bit see 6.12.3.

Table 4 — Register behaviors for a service action received on a registered I_T nexus

Service action [c] [d]	RESERVATION KEY field = I_T nexus reservation key [a]	SERVICE ACTION RESERVATION KEY field	SPEC_I_PT [b]	Result
REGISTER	no	ignore	ignore	Return RESERVATION CONFLICT status
	yes	zero	zero	Unregister the I_T nexus (see 5.6.10.3)
			one	Return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.
		non-zero	zero	Change the reservation key of the I_T nexus to the value specified in the SERVICE ACTION RESERVATION KEY field.
			one	Return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.
REGISTER AND IGNORE EXISTING KEY	ignore	zero	zero	Unregister the I_T nexus (see 5.6.10.3)
			one	Return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.
		non-zero	zero	Change the reservation key of the I_T nexus to the value specified in the SERVICE ACTION RESERVATION KEY field.
			one	Return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.
~~REGISTER AND MOVE~~	~~no~~	~~ignore~~	~~ignore~~	~~Return RESERVATION CONFLICT status~~
	~~yes~~	~~zero~~		~~Return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.~~
		~~non-zero~~ [f]		~~See 5.6.7 for the registration and the move specifications.~~

[a] Does the RESERVATION KEY field contain the reservation key of the I_T nexus being used for the PERSISTENT RESERVE OUT command?

[b] If the SPEC_I_PT bit is set to zero the device server shall ignore the PERSISTENT RESERVE OUT commands additional parameter data (see 6.12.3).

[c] APTPL bit may be set to one.

[d] For usage of the ALL_TG_PT bit see 6.12.3.

~~[e] If the TransportID is not valid then return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN PARAMTER LIST.~~

~~[f] The application client should use the same reservation key as the backup application.~~

~~If a PERSISTENT RESERVE OUT command with REGISTER AND IGNORE EXISTING KEY service action is sent when an established registration exists, that registration shall be superseded with the specified service action reservation key. If a PERSISTENT RESERVE OUT command with REGISTER AND IGNORE EXISTING KEY service action is sent when there is no established registration, a new registration shall be established.~~

If a PERSISTENT RESERVE OUT command with a REGISTER service action or REGISTER AND IGNORE EXISTING KEY service action, ~~or REGISTER AND MOVE 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 code shall be set to INSUFFICIENT REGISTRATION RESOURCES.

In response to a PERSISTENT RESERVE OUT command with a REGISTER service action or REGISTER AND IGNORE EXISTING KEY service action, ~~or REGISTER AND MOVE 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 SCOPE and TYPE fields;
d) Map the reservation key to the ~~registering~~ I_T nexus being registered using the indication of the target port associated with the registration and either the initiator port name (see 3.1.49) on SCSI transport protocols where port names are required or the initiator port identifier (see 3.1.48) on SCSI transport protocols where port names are not required;
e) Register the reservation key without changing any persistent reservation that may exist; and
f) Retain the reservation key and associated information.

After the registration request has been processed, the device server shall then allow other PERSISTENT RESERVE OUT commands from the registered I_T nexus to be processed. ~~For each I_T nexus that is the source a PERSISTENT RESERVE OUT command with a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY 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 or the REGISTER AND IGNORE EXISTING KEY service action from the same I_T nexus~~ or ~~until the registration is~~ removed (see 5.6.10).

Any PERSISTENT RESERVE OUT command service action received from an unregistered I_T nexus, other than the REGISTER or the REGISTER AND IGNORE EXISTING KEY service action, shall be rejected with a RESERVATION CONFLICT status.

It is not an error for an I_T nexus that is registered to be registered again with the same reservation key or a new reservation key. A registration shall have no effect on any other registrations (e.g., when more than one I_T nexus is registered with the same reservation key and one of those I_T nexuses registers again it has no effect on the other I_T nexus' registrations). A registration that contains a non-zero value in the SERVICE ACTION RESERVATION KEY field shall have no effect on any persistent reservations (i.e., the reservation key for an I_T nexus may be changed without affecting any previously created persistent reservation).

Multiple I_T nexuses may be registered with the same reservation key. An application client may use the same reservation key for other I_T nexuses and logical units.

5.6.7 Registering and moving the reservation

The PERSISTENT RESERVE OUT command REGISTER AND MOVE service action is used to register a specified I_T nexus (see table 5) and move the reservation to that I_T nexus. ~~from one I_T nexus to another.~~

Table 5 — Register behaviors for a REGISTER AND MOVE service action

Service action [a b c]	RESERVATION KEY field = I_T nexus reservation key [d]	SERVICE ACTION RESERVATION KEY field	UNREG	Result
received on an unregistered I_T nexus	ignore	ignore	ignore	Return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.
received on a registered I_T nexus	no	ignore	ignore	Return RESERVATION CONFLICT status
	yes	zero		Return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB.
		non-zero [e]	zero	The I_T nexus that the PERSISTENT RESERVE OUT command is associated with shall remain registered. See this subclause for the registration and the move specifications.
			one	On completion of the registration and move the I_T nexus on which the PERSISTENT RESERVE OUT command was received shall become unregistered. See this subclause for the registration and the move specifications.

[a] APTPL bit may be set to one.
[b] The SPEC_I_PT bit shall be ignored.
[c] For usage of the ALL_TG_PT bit see 6.12.3.
[d] Does the RESERVATION KEY field contain the reservation key of the I_T nexus being used for the PERSISTENT RESERVE OUT command?
[e] The application client should use the same reservation key as the backup application.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE 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 code shall be set to INSUFFICIENT REGISTRATION RESOURCES.

In addition to the requirements in table 5 the reservation and move shall only occur if:

a) the TransportID is different than the initiator port of the I_T nexus used for this command; and
b) the persistent reservation is not of the type Write Exclusive - All Registrants or Exclusive Access - All Registrants.

If the type of reservation is all registrants, then the PERSISTENT RESERVE OUT shall be rejected with a RESERVATION CONFLICT status. ~~the I_T nexus still remains a reservation holder. If the type of reservation is registrants only, then the I_T nexus still maintains access.~~

If the TransportID is the same as the initiator port of the I_T nexus used for this command then return a CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN PARAMETER LIST.

In response to a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action the device server shall perform a registration and move 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 SCOPE and TYPE fields;
d) Map the reservation key to the ~~registering~~ I_T nexus being registered using the indication of the target port associated with the registration and either the initiator port name (see 3.1.49) on SCSI transport protocols where port names are required or the initiator port identifier (see 3.1.48) on SCSI transport protocols where port names are not required;
e) Register the reservation key without changing any persistent reservation that may exist;
f) Retain the reservation key and associated information.
g) Register the specified I_T nexus by copying the RESERVATION KEY field to the I_T nexus pointed to by the TransportID and the RELATIVE TARGET PORT field (see 7.6.4.6xx);
h) Release the persistent reservation for the persistent reservation holder; and
i) Establish a persistent reservation for the specified I_T nexus using the same scope and type as the persistent reservation being moved.
j) If the UNREG bit is set to one the I_T nexus on which PERSISTENT RESERVE OUT command was received shall become unregistered.

~~The reservation and move shall occur if:~~

~~a) the SERVICE ACTION RESERVATION KEY field is non-zero (i.e., it does not specify an unregistration);~~
~~b) the I_T nexus used for the PERSISTENT RESERVE OUT command is a reservation holder and the RESERVATION KEY field is set to the reservation key of that I_T nexus;~~
~~c) the TransportID is different than the initiator port of the I_T nexus used for this command; and~~
~~d) the persistent reservation is not of the type Write Exclusive - All Registrants or Exclusive Access - All Registrants.~~
~~e) a single I_T nexus is specified that is different than the I_T nexus used for the PERSISTENT RESERVE OUT command (i.e., the spec_i_pt bit is set to one, the additional parameter data contains exactly one TransportID, the TransportID does not describe the initiator port sending the command, and the all_tg_pt bit is set to zero or there is only one target port);~~

~~The device server shall perform the register and move these actions in the following an uninterrupted sequence:~~

~~1) register the specified I_T nexus by copying the RESERVATION KEY field to the I_T nexus pointed to by the TransportID and the RELATIVE TARGET PORT field (see 7.6.4.6xx);~~
~~2) release the persistent reservation for the persistent reservation holder remove the persistent reservation from the I_T nexus used for the PERSISTENT RESERVE OUT command; and~~
~~3) establish a persistent reservation for the specified I_T nexus using the same scope and type as the previous persistent reservation being moved.~~

~~If any of the conditions in this subclause are not met, the command shall be rejected with:~~

~~a) RESERVATION CONFLICT status; or~~
~~b) CHECK CONDITION status with a sense key of ILLEGAL REQUEST and an additional sense code of INVALID FIELD IN CDB or INVALID FIELD IN PARAMETER LIST,~~

~~as described in 5.6.5.~~

5.6.8 Reserving

An application client creates a persistent reservation by issuing a PERSISTENT RESERVE OUT command with RESERVE service action through a registered I_T nexus with the following parameters:

a) RESERVATION KEY set to the value of the reservation key that is registered for the I_T_L nexus; and
b) TYPE and SCOPE fields set to the persistent reservation being created.

Only one persistent reservation is allowed at a time per logical unit and that persistent reservation has a scope of LU_SCOPE.

If the device server receives a PERSISTENT RESERVE OUT command from an I_T nexus other than a persistent reservation holder (see 5.6.9) that attempts to create a persistent reservation when a persistent reservation already exists for the logical unit, then the command shall be rejected with a RESERVATION CONFLICT status.

If a persistent reservation holder attempts to modify the TYPE or SCOPE of an existing persistent reservation, then the command shall be rejected with a RESERVATION CONFLICT status.

If the device server receives a PERSISTENT RESERVE OUT command with RESERVE service action where the TYPE and SCOPE are the same as the existing TYPE and SCOPE from a persistent reservation holder, it shall not make any change to the existing persistent reservation and shall return a GOOD status.

See 5.6.1 for information on when a persistent reservation takes effect.

5.6.9 Persistent reservation holder

The persistent reservation holder is determined by the type of the persistent reservation as follows:

a) For a persistent reservation of the type Write Exclusive – All Registrants or Exclusive Access – All Registrants, the persistent reservation holder is any registered I_T nexus; or
b) For all other persistent reservation types, the persistent reservation holder is the I_T nexus for which the reservation was established ~~from which the~~ with a PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action, RESERVE service action, PREEMPT service action, or PREEMPT AND ABORT service actions ~~was received~~.

A persistent reservation holder has its reservation key returned in the parameter data from a PERSISTENT RESERVE IN command with READ RESERVATION service action as follows:

a) For a persistent reservation of the type Write Exclusive – All Registrants or Exclusive Access – All Registrants, the reservation key shall be set to zero; or
b) For all other persistent reservation types, the reservation key shall be set to the registered reservation key for the I_T nexus that holds the persistent reservation.

It is not an error for a persistent reservation holder to send a PERSISTENT RESERVE OUT command with RESERVE service action to the reserved logical unit with TYPE and SCOPE fields that match those of the persistent reservation (see 5.6.8).

A persistent reservation holder is allowed to release the persistent reservation using the PERSISTENT RESERVE OUT command with RELEASE service action.

If the registration of the persistent reservation holder is removed (see 5.6.10.1.1), the reservation is automatically released. When the persistent reservation holder is more than one I_T nexus, the reservation is not automatically released until the registrations for all persistent reservation holder I_T nexuses are removed.

5.6.10 Releasing persistent reservations and removing registrations

5.6.10.1 Overview

5.6.10.1.1 Summary of service actions that release persistent reservations and remove registrations

[no changes]

5.6.10.1.2 Processing for released registrants only persistent reservations

[no changes]

5.6.10.1.3 Processing for released all registrants persistent reservations

[no changes]

5.6.10.1.4 Processing for other released persistent reservations

[no changes]

5.6.10.2 Releasing

[no changes]

5.6.10.3 Unregistering

[no changes]

5.6.10.4 Preempting

5.6.10.4.1 Overview

[no changes]

5.6.10.4.2 Failed persistent reservation preempt

[no changes]

5.6.10.4.3 Preempting persistent reservations and registration handling

An application client may preempt the persistent reservation with another persistent reservation by issuing a PERSISTENT RESERVE OUT command with PREEMPT service action or PREEMPT AND ABORT service action through a registered I_T nexus with the following parameters:

a) RESERVATION KEY field set to the value of the reservation key that is registered for the I_T_L nexus;
b) SERVICE ACTION RESERVATION KEY field set to the value of the reservation key of the persistent reservation to be preempted; and
c) TYPE and SCOPE fields set to define a new persistent reservation. The SCOPE and TYPE of the persistent reservation created by the preempting I_T nexus may be different than those of the persistent reservation being preempted.

If the SERVICE ACTION RESERVATION KEY field identifies a persistent reservation holder (see 5.6.9), the device server shall perform a preempt by doing the following as an uninterrupted series of actions:

a) Release the persistent reservation for the holder identified by the SERVICE ACTION RESERVATION KEY field;
b) Remove the registrations for all I_T nexuses identified by the SERVICE ACTION RESERVATION KEY field, except the I_T nexus that is being used for the PERSISTENT RESERVE OUT command. If an all registrants persistent reservation is present and the SERVICE ACTION RESERVATION KEY field is set to zero then all registrations shall be removed except for that of the I_T nexus that is being used for the PERSISTENT RESERVE OUT command;
c) Establish a persistent reservation for the preempting I_T nexus using the contents of the SCOPE and TYPE fields;
d) Process tasks as defined in 5.6.1; and

e) Establish a unit attention condition for every initiator port associated with every I_T nexus that lost its persistent reservation and/or registration. The sense key shall be set to UNIT ATTENTION and the additional sense code shall be set to REGISTRATIONS PREEMPTED.

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 I_T nexus.

The following tasks shall be subjected in a vendor specific manner either to the restrictions established by the persistent reservation being preempted or to the restrictions established by the preempting I_T nexus:

a) A task received after the arrival, but before the completion of the PERSISTENT RESERVE OUT command with the PREEMPT service action or the PREEMPT AND ABORT service action; or
b) A task in the dormant, blocked, or enable state at the time the PERSISTENT RESERVE OUT command with the PREEMPT service action or the PREEMPT AND ABORT service action is received.

Completion status shall be returned for each task unless it was aborted with PREEMPT AND ABORT service action and TAS bit set to zero (see 7.4.6).

A PERSISTENT RESERVE OUT with a PREEMPT service action or a PREEMPT AND ABORT service action with the SERVICE ACTION RESERVATION KEY value equal to the persistent reservation holder's reservation key is not an error. In that case the device server shall establish the new persistent reservation and maintain the registration.

5.6.10.4.4 Removing registrations

[no changes]

5.6.10.5 Preempting and aborting

The application client's request for and the device server's responses to a PERSISTENT RESERVE OUT command PREEMPT AND ABORT service action are identical to the responses to a PREEMPT service action (see 5.6.10.4) except for the following additions. If no reservation conflict occurred, the device server shall perform the following uninterrupted series of actions:

a) If the persistent reservation is not an all registrants type then:
   A) If the TST field is 000b (see 7.4.6) and an ACA condition exists for initiator ports other than the initiator port associated with the persistent reservation being preempted, the PERSISTENT RESERVE OUT command shall be terminated prior to processing with a status of ACA ACTIVE if the NACA bit equals one in the CDB CONTROL byte (see SAM-2) or BUSY if the NACA bit equals zero; or
   B) If the TST field contains 001b, then the ACA condition for initiator ports other than the initiator port associated with the persistent reservation being preempted shall not prevent the processing of the PERSISTENT RESERVE OUT command;
b) Perform the uninterrupted series of actions described for the PREEMPT service action (see 5.6.10.4);
c) All tasks from the initiator port or initiator ports associated with the persistent reservations being preempted (called preempted tasks) except the task containing the PERSISTENT RESERVE OUT command itself shall be terminated. Application client notification shall be provided, as specified by the TAS bit in the Control mode page (see 7.4.6) that applies to the initiator port associated with the persistent reservation being preempted (called the preempted initiator port), as follows:
   A) If the TAS bit is set to zero then all preempted tasks shall be terminated as if an ABORT TASK SET task management function had been performed by each preempted initiator port; or
   B) If the TAS bit is set to one then all preempted tasks from initiator ports other than the initiator port that sent the PREEMPT AND ABORT service action shall be terminated with a TASK ABORTED status (see SAM-2). Any preempted tasks from the initiator port that sent the PREEMPT AND ABORT service action shall be terminated as if an ABORT TASK SET task management function had been performed by that initiator port.

   If a terminated task is a command that causes the device server to generate additional commands and data transfers (e.g., EXTENDED COPY), all commands and data transfers generated by the command shall be terminated before the ABORT TASK SET task management function is considered completed.

After the ABORT TASK SET function has completed, all new tasks are subject to the persistent reservation restrictions established by the preempting initiator port;

d) If the persistent reservation is not an all registrants type then the device server shall clear any ACA condition associated with an initiator port being preempted and shall clear any tasks with an ACA attribute from that initiator port;
e) If the persistent reservation is an all registrants type then the device server shall clear any ACA condition and shall clear any tasks with an ACA attribute; and
f) For logical units that implement the PREVENT ALLOW MEDIUM REMOVAL command, the device server shall perform an action equivalent to the processing of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field equal to zero for the initiator port or initiator ports associated with the persistent reservation being preempted (see 6.13).

The actions described in this subclause shall be performed for all I_T nexuses that are registered with the SERVICE ACTION RESERVATION KEY value, without regard for whether the preempted I_T nexuses hold the persistent reservation. If an all registrants persistent reservation is present the device server shall abort all tasks for all registered I_T nexuses.

5.6.10.6 Clearing

[no changes]

6.11 PERSISTENT RESERVE IN command

6.11.1 PERSISTENT RESERVE IN command description

[no changes]

6.11.2 PERSISTENT RESERVE IN service actions

6.11.2.1 Summary of PERSISTENT RESERVE IN service actions

[no changes]

6.11.2.2 READ KEYS

[no changes]

6.11.2.3 READ RESERVATION

[no changes]

6.11.2.4 REPORT CAPABILITIES

[no changes]

6.11.3 PERSISTENT RESERVE IN parameter data for READ KEYS

[no changes]

6.11.4 PERSISTENT RESERVE IN parameter data for READ RESERVATION

6.11.4.1 Format of PERSISTENT RESERVE IN parameter data for READ RESERVATION

[no changes]

6.11.4.2 Persistent reservations Scope

[no changes]

6.11.4.3 Persistent Reservations Type

[no changes]

6.11.5 PERSISTENT RESERVE IN parameter data for REPORT CAPABILITIES

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the REPORT CAPABILITIES service action is shown in table 6.

Table 6 — PERSISTENT RESERVE IN parameter data for REPORT CAPABILITIES

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1		LENGTH (0008h)						(LSB)
2	Reserved			CRH	SIP_C	ATP_C	Reserved	PTPL_C
3	TMV	Reserved						PTPL_A
4		PERSISTENT RESERVATION TYPE MASK						
5								
6		Reserved						
7								

The LENGTH field indicates the length in bytes of the parameter data. If the ALLOCATION LENGTH field in the CDB is too small to transfer all of the parameter data, the length shall not be adjusted to reflect the truncation.

~~A MOVE (REGISTER AND MOVE service action supported) bit set to one indicates that the device server supports the REGISTER AND MOVE service action. A MOVE bit set to zero indicates the device server does not support the REGISTER AND MOVE service action.~~

~~A PRE_SIP_C (Specify Initiator Ports Capable) bit set to one indicates that the device server supports the SPEC_I_PT bit in the PERSISTENT RESERVE OUT command parameter data (see 6.12.3) for the PREEMPT service action and the PREEMPT AND ABORT service action. A PRE_SIP_C bit set to zero indicates that the device server does not support the SPEC_I_PT bit in the PERSISTENT RESERVE OUT command parameter data for the PREEMPT service action and the PREEMPT AND ABORT service action.~~

~~A PRE_ATP_C (All Target Ports Capable) bit set to one indicates that the device server supports the ALL_TG_PT bit in the PERSISTENT RESERVE OUT command parameter data for the PREEMPT service action and the PREEMPT AND ABORT service action. A PRE_ATP_C bit set to zero indicates that the device server does not support the ALL_TG_PT bit in the PERSISTENT RESERVE OUT command parameter data for the PREEMPT service action and the PREEMPT AND ABORT service actions.~~

A CRH (Compatible Reservation Handling) bit set to one indicates that the device server supports the exceptions to the SPC-2 RESERVE and RELEASE commands described in 5.6.2 A CRH bit set to zero indicates that RESERVE(6) command, RESERVE(10) command, RELEASE(6) command, and RELEASE(10) command are processed as defined in SPC-2.

An SIP_C (Specify Initiator Ports Capable) bit set to one indicates that the device server supports the SPEC_I_PT bit in the PERSISTENT RESERVE OUT command parameter data (see 6.12.3) ~~for the REGISTER service action.~~

~~REGISTER AND IGNORE EXISTING KEY service action, and REGISTER AND MOVE service actions (if supported)~~ An SIP_C bit set to zero indicates that the device server does not support the SPEC_I_PT bit in the PERSISTENT RESERVE OUT command parameter data~~ for the REGISTER service action, REGISTER AND IGNORE EXISTING KEY service action, and REGISTER AND MOVE service actions (if supported)~~.

An ATP_C (All Target Ports Capable) bit set to one indicates that the device server supports the ALL_TG_PT bit in the PERSISTENT RESERVE OUT command parameter data~~ for the REGISTER service action, REGISTER AND IGNORE EXISTING KEY service action, and REGISTER AND MOVE service actions (if supported)~~. An ATP_C bit set to zero indicates that the device server does not support the ALL_TG_PT bit in the PERSISTENT RESERVE OUT command parameter data~~ for the REGISTER service action, REGISTER AND IGNORE EXISTING KEY service action, and REGISTER AND MOVE service actions (if supported)~~.

.....

6.12 PERSISTENT RESERVE OUT command

6.12.1 PERSISTENT RESERVE OUT command introduction

The PERSISTENT RESERVE OUT command (see table 7) is used to request service actions that reserve a logical unit or element for the exclusive or shared use of a particular I_T nexus. The command uses other service actions to manage and remove such persistent reservations.

I_T nexuses performing PERSISTENT RESERVE OUT service actions are identified by a registered reservation key provided by the application client. An application client may use the PERSISTENT RESERVE IN command to obtain the reservation key, if any, for the I_T nexus holding a persistent reservation and may use the PERSISTENT RESERVE OUT command to preempt that persistent reservation.

Table 7 — PERSISTENT RESERVE OUT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (5Fh)							
1	Reserved				SERVICE ACTION			
2	SCOPE				TYPE			
3	Reserved							
4	Reserved							
5	Reserved							
6	Reserved							
7	(MSB)			PARAMETER LIST LENGTH				
8								(LSB)
9	CONTROL							

If a PERSISTENT RESERVE OUT command 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 code shall be set to INSUFFICIENT REGISTRATION RESOURCES.

The PERSISTENT RESERVE OUT command contains fields that specify a persistent reservation service action, the intended scope of the persistent reservation, and the restrictions caused by the persistent reservation. The TYPE and SCOPE fields are defined in 6.11.4.2 and 6.11.4.3. If a SCOPE field specifies a scope that is not imple-

mented, the device server shall return a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and additional sense code shall be set to INVALID FIELD IN CDB.

Fields contained in the PERSISTENT RESERVE OUT parameter list specify the information required to perform a particular persistent reservation service action.

If the SPEC_I_PT bit (see 6.12.3) is zero, the parameter list shall be 24 bytes in length and the PARAMETER LIST LENGTH field shall contain 24 (18h). If the SPEC_I_PT bit is set to zero and the parameter list length is not 24, the device server shall return a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to PARAMETER LIST LENGTH ERROR. If the SPEC_I_PT bit is set to one, the PARAMETER LIST LENGTH field specifies the number of bytes of parameter data for the PERSISTENT RESERVE OUT command.

6.12.2 PERSISTENT RESERVE OUT service actions

When processing the PERSISTENT RESERVE OUT service actions, the device server shall increment the PRgeneration value as specified in 6.11.3.

The PERSISTENT RESERVE OUT command service actions are defined in table 8.

Table 8 — PERSISTENT RESERVE OUT service action codes

Code	Name	Description	PRGENERATION field incremented (see 6.11.3)	Reference
00h	REGISTER	Register a reservation key with the device server (see 5.6.5) or unregister a reservation key (see 5.6.8.3).	Yes	6.12.3
01h	RESERVE	Creates a persistent reservation having a specified SCOPE and TYPE (see 5.6.6). The SCOPE and TYPE of a persistent reservation are defined in 6.11.4.2 and 6.11.4.3.	No	6.12.3
02h	RELEASE	Releases the selected persistent reservation (see 5.6.8.2).	No	6.12.3
03h	CLEAR	Clears all reservation keys (i.e., registrations) and all persistent reservations (see 5.6.8.6).	Yes	6.12.3
04h	PREEMPT	Preempts persistent reservations and/or removes registrations (see 5.6.8.4).	Yes	6.12.3
05h	PREEMPT AND ABORT	Preempts persistent reservations and/or removes registrations and aborts all tasks for all preempted I_T nexuses (see 5.6.8.4 and 5.6.8.5).	Yes	6.12.3
06h	REGISTER AND IGNORE EXISTING KEY	Register a reservation key with the device server (see 5.6.5) or unregister a reservation key (see 5.6.8.3).	Yes	6.12.3
07h	REGISTER AND MOVE	Register a reservation key with the device server for another I_T nexus and move a persistent reservation to that I_T nexus (see 5.6.6 and see 5.6.7)	yes	6.12.4
07h - 1Fh	Reserved			

The parameter list values for each service action are specified in 6.12.3.

6.12.3 Basic PERSISTENT RESERVE OUT parameter list

The parameter list required to perform the following PERSISTENT RESERVE OUT commands service actions is defined in table 9:

a) REGISTER service action;
b) RESERVE service action;
c) RELEASE service action;
d) CLEAR service action;
e) PREEMPT service action;
f) PREEMPT AND ABORT service action; and
g) REGISTER AND IGNORE EXISTING KEY service action.

All fields shall be sent in the specified ~~on all~~ PERSISTENT RESERVE OUT command service actions, even if the field is not required for the specified service action and scope values

....

6.12.4 PERSISTENT RESERVE OUT REGISTER AND MOVE service action parameter list

The parameter list required to perform the PERSISTENT RESERVE OUT command REGISTER AND MOVE service action is defined in table 9.

Table 9 — PERSISTENT RESERVE OUT REGISTER AND MOVE service action parameter list

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)			RESERVATION KEY				
7								(LSB)
8	(MSB)			SERVICE ACTION RESERVATION KEY				
15								(LSB)
16	Reserved							
17	Reserved						UNREG	APTPL
18	(MSB)			RELATIVE TARGET PORT IDENTIFIER				
19								(LSB)
20	(MSB)			TRANSPORTID PARAMETER DATA LENGTH (n - 23)				
23								(LSB)
24				TransportID				
n								

The RESERVATION KEY field contains an 8-byte value provided by the application client to the device server to identify the I_T nexus that is the source of the PERSISTENT RESERVE OUT command. The device server shall verify that the contents of the RESERVATION KEY field in a PERSISTENT RESERVE OUT command parameter data matches the registered reservation key for the I_T nexus from which the task was received.

When a PERSISTENT RESERVE OUT command specifies a RESERVATION KEY field other than the reservation key registered for the I_T nexus the device server shall return a RESERVATION CONFLICT status. The reservation key of the I_T nexus shall be verified to be correct regardless of the SERVICE ACTION and SCOPE field values.

The SERVICE ACTION RESERVATION KEY field contains the reservation key to be registered to the specified I_T nexus.

The APTPL (Activate Persist Through Power Loss) bit set to one is optional. If a device server that does not support an APTPL bit set to one receives that value it shall return a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and additional sense code shall be set to INVALID FIELD IN PARAMETER LIST.

If the last valid APTPL bit value received by the device server is zero, the loss of power in the SCSI target device shall release the persistent reservation for the logical unit and remove all registered reservation keys (see 5.6.5). If the last valid APTPL bit value received by the device server is one, the logical unit shall retain any persistent reservation(s) that may be present and all reservation keys (i.e., registrations) for all I_T nexuses even if power is lost and later returned (see 5.6.3).

The UNREG (unregister) bit set to zero specifies that the device server shall not unregister the I_T nexus on which the PERSISTENT RESERVE OUT command REGISTER AND MOVE service action was received. An UNREG bit set to one specifies that the device server shall unregister the I_T nexus on which the PERSISTENT RESERVE OUT command REGISTER AND MOVE service action was received.

The RELATIVE TARGET PORT IDENTIFIER field (see 7.6.4.6) contains the relative target port identifier of the SCSI target port to which the persistent reservation applies.

The TRANSPORTID DESCRIPTOR LENGTH field indicates the number of bytes of the TransportID descriptor that follow, shall be a minimum of 24 bytes, and shall be a multiple of 4.

The format of a TransportID descriptor is specified in 7.5.4.

The command shall be terminated with a CHECK CONDITION status and 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 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.

6.12.5 Descriptor type codes

Target descriptors and segment descriptors share a single set of code values that identify the type of descriptor (see table 10). Segment descriptors use codes in the range 00h to BFh. The definitions of codes between C0h and DFh are vendor specific. Target descriptors use codes in the range E0h to FFh.

Table 10 — EXTENDED COPY descriptor type codes (part 1 of 2)

Descriptor type code	Reference	Description [a]	Shorthand [a]
00h	6.3.7.3	Copy from block device to stream device	block<arrow(symbol)>Æ stream
01h	6.3.7.4	Copy from stream device to block device	stream<arrow(symbol)>Æ block
02h	6.3.7.5	Copy from block device to block device	block<arrow(symbol)>Æ block
03h	6.3.7.6	Copy from stream device to stream device	stream<arrow(symbol)>Æ stream
04h	6.3.7.7	Copy inline data to stream device	inline<arrow(symbol)>Æ stream
05h	6.3.7.8	Copy embedded data to stream device	embedded<arrow(symbol)>Æ stream
06h	6.3.7.9	Read from stream device and discard	stream<arrow(symbol)>Æ discard
07h	6.3.7.10	Verify block or stream device operation	
08h	6.3.7.11	Copy block device with offset to stream device	block<o><arrow(symbol)>Æ stream
09h	6.3.7.12	Copy stream device to block device with offset	stream<arrow(symbol)>Æ block<o>
0Ah	6.3.7.13	Copy block device with offset to block device with offset	block<o><arrow(symbol)>Æ block<o>
0Bh	6.3.7.3	Copy from block device to stream device and hold a copy of processed data for the application client [b]	block<arrow(symbol)>Æ stream +application client
0Ch	6.3.7.4	Copy from stream device to block device and hold a copy of processed data for the application client [b]	stream<arrow(symbol)>Æ block +application client
0Dh	6.3.7.5	Copy from block device to block device and hold a copy of processed data for the application client [b]	block<arrow(symbol)>Æ block +application client

[a] Block devices are those with peripheral device type codes 0h (i.e, direct-access), 4h (i.e., write-once), 5h (i.e., CD/DVD), 7h (i.e., optical memory), and Eh (i.e., simplified direct-access). Stream devices are those devices with peripheral device type codes 1h (i.e., sequential-access) and 3h (i.e., processor). Sequential-access stream (indicated by "tape" in the shorthand column) devices are those with peripheral device type code 1h. See 6.4.2 for peripheral device type code definitions.

[b] The application client shall use the RECEIVE COPY RESULTS with a RECEIVE DATA service action to retrieve data held for it by the copy manager (see 6.17.3).

Table 10 — EXTENDED COPY descriptor type codes (part 2 of 2)

Descriptor type code	Reference	Description [a]	Shorthand [a]
0Eh	6.3.7.6	Copy from stream device to stream device and hold a copy of processed data for the application client [b]	stream<arrow(symbol)>Æ stream +application client
0Fh	6.3.7.9	Read from stream device and hold a copy of processed data for the application client [b]	stream<arrow(symbol)>Æ discard +application client
10h	6.3.7.14	Write filemarks to sequential-access device	filemark<arrow(symbol)>Æ tape
11h	6.12.5.1	Space records or filemarks on sequential-access device	space<arrow(symbol)>Æ tape
12h	6.3.7.16	Locate on sequential-access device	locate<arrow(symbol)>Æ tape
13h	6.3.7.17	Image copy from sequential-access device to sequential-access device	<i>tape<arrow(symbol)>Æ <i>tape
14h	6.3.7.18	Register persistent reservation key	
15h	6.12.5.1	Source third party persistent reservations I_T nexus	
16h - BFh		Reserved for segment descriptors	
C0h - DFh		Vendor unique descriptors	
E0h	7.5.3.2	Fibre Channel World Wide Name target descriptor	
E1h	7.5.3.3	Fibre Channel N_Port target descriptor	
E2h	7.5.3.4	Fibre Channel N_Port with World Wide Name checking target descriptor	
E3h	7.5.3.5	Parallel Interface T_L target descriptor	
E4h	6.3.6.2	Identification descriptor target descriptor	
E5h	7.5.3.8	IPv4 target descriptor	
E6h	6.3.6.3	Alias target descriptor	
E7h	7.5.3.7	RDMA target descriptor	
E8h	7.5.3.6	IEEE 1394 EUI-64 target descriptor	
E9h	7.5.3.9	SAS Serial SCSI Protocol target descriptor	
EAh - FFh		Reserved for target descriptors	

[a] Block devices are those with peripheral device type codes 0h (i.e, direct-access), 4h (i.e., write-once), 5h (i.e., CD/DVD), 7h (i.e., optical memory), and Eh (i.e., simplified direct-access). Stream devices are those devices with peripheral device type codes 1h (i.e., sequential-access) and 3h (i.e., processor). Sequential-access stream (indicated by "tape" in the shorthand column) devices are those with peripheral device type code 1h. See 6.4.2 for peripheral device type code definitions.

[b] The application client shall use the RECEIVE COPY RESULTS with a RECEIVE DATA service action to retrieve data held for it by the copy manager (see 6.17.3).

6.12.5.1 Source third party persistent reservations I_T nexus

The segment descriptor format shown in table 11 instructs the copy manager to issue a PERSISTENT RESERVATION OUT command REGISTER AND MOVE service action with the specified I_T nexus after all other segment descriptors have been processed. If an error is detected any time after receiving a source third party persistent

reservation I_T nexus segment descriptor the PERSISTENT RESERVATION OUT command REGISTER AND MOVE service action shall be processed before status is returned for the EXTENDED COPY command.

This segment descriptor should be placed at or near the beginning of the list of segment descriptors to assure the copy manager processes the PERSISTENT RESERVATION OUT command REGISTER AND MOVE service action in the event of an error that terminates the processing of segment descriptors. If an error is detected in a segment descriptor and this segment descriptor has not been processed the PERSISTENT RESERVATION OUT command REGISTER AND MOVE service action may not occur.

Placing more than one source third party persistent reservations I_T nexus segment descriptor in the list of descriptors is not an error.

Any source third party persistent reservations I_T nexus segment descriptors shall be processed after all other segment descriptors have been processed.

Table 11 — Source third party persistent reservations I_T nexus segment descriptor

Bit / Byte	7	6	5	4	3	2	1	0
0	DESCRIPTOR TYPE CODE (15h)							
1	Reserved							
2	(MSB)			DESCRIPTOR LENGTH (n-3)				
3								(LSB)
4	Reserved							
5	Reserved							
6	(MSB)			DESTINATION TARGET DESCRIPTOR INDEX				
7								(LSB)
8	(MSB)			RESERVATION KEY				
15								(LSB)
16	(MSB)			SERVICE ACTION RESERVATION KEY				
23								(LSB)
24	Reserved							
25	Reserved						UNREG	APTPL
26	(MSB)			RELATIVE TARGET PORT IDENTIFIER				
27								(LSB)
28	(MSB)			TRANSPORTID PARAMETER DATA LENGTH (n - 31)				
31								(LSB)
32				TransportID				
n								

The DESCRIPTOR TYPE CODE field is described in 6.12.5 and 6.3.7.1. Descriptor type code 15h instructs the copy manager to send PERSISTENT RESERVATION OUT command REGISTER AND MOVE service action, with the contents of bytes 8 through n as the PERSISTENT RESERVE OUT REGISTER AND MOVE service action parameter list, to the target port identified by the DESTINATION TARGET DESCRIPTOR INDEX field. If the PERIPHERAL DEVICE TYPE field in the target descriptor identified by the DESTINATION TARGET DESCRIPTOR INDEX field does not contain 01h, the copy manager shall terminate the command with a CHECK CONDITION status. The sense key shall be set to COPY ABORTED and the additional sense code shall be set to INVALID OPERATION FOR COPY SOURCE OR DESTINATION.

The DESCRIPTOR LENGTH field shall contain the length in bytes of the fields that follow the DESCRIPTOR LENGTH field. The value in the DESCRIPTOR LENGTH field shall be a multiple of 4.

For a description of the RESERVATION KEY field, SERVICE ACTION RESERVATION KEY field, UNREG bit, APTPL bit, RELATIVE TARGET PORT IDENTIFIER field, TRANSPORTID DESCRIPTOR LENGTH field, and the TransportID see 6.12.4.

~~The RELATIVE TARGET PORT IDENTIFIER specifies the target port that the copy manager shall use to restore the I_T nexus of the persistent reservation. The copy manager shall place this value in the RELATIVE TARGET PORT IDENTIFIER field of the PERSISTENT RESERVATION OUT command REGISTER AND MOVE service action.~~

~~The TRANSPORTID DESCRIPTOR LENGTH field indicates the number of bytes of the TransportID descriptor that follow. shall be a minimum of 24 bytes, and shall be a multiple of 4.~~

~~The TransportID specifies the initiator port that the copy manager shall use to restore the I_T nexus of the persistent reservation. The copy manager shall place this value in the TransportID field of the PERSISTENT RESERVATION OUT command REGISTER AND MOVE service action.~~

7.5.4.1 Overview of TransportID identifiers

An application client may use a TransportID to identify an initiator port other than the initiator port that is transporting the command and parameter data.

~~TransportIDs are initiator port identifiers used as:~~

~~a) Access identifiers (see 8.3.1.3.2) in ACL ACEs to allow access to a logical unit from an initiator port; and~~
~~b) Initiator port identifiers in the additional parameter data (see 6.12.3) for the PERSISTENT RESERVE OUT command with REGISTER or REGISTER AND IGNORE EXISTING KEY service action if the SPEC_I_PT bit is set to one.~~