To: T10 Technical Committee
From: Rob Elliott, Compaq Computer Corporation (Robert.Elliott@compaq.com)
Date: 13 March 2002
Subject: T10/02-065r2 SPC-3 Persistent reservations corrections

Revision History
Revision 0 (23 February 2002) first revision
Revision 1 (7 March 2002) changed REGISTRATIONS RELEASED (non-existent) to REGISTRATIONS PREEMPTED in issue #1 backgrounder; added nexus in 5.5.3.4; clarified "the reservation key" in 5.5.3.7.3.3
Revision 2 (13 March 2002) incorporated feedback from the March 2002 CAP WG.

Related Documents
spc3r04 - SCSI Primary Commands - 3 revision 4 (Ralph Weber)
01-099r6 SPC-3 Letting persistent reservations ignore target ports (Rob Elliott)
01-204r2 SPC-3 Proposal for an additional persistent Reservations type (Roger Cummings)

Overview
Numerous technical and editorial issues in the persistent reservations clause are addressed.

Issue #1: Which additional sense code for UA from PREEMPT or PREEMPT AND ABORT
SPC-3 provides conflicting rules on which additional sense code is returned when a persistent reservation is preempted - RESERVATIONS RELEASED or REGISTRATIONS PREEMPTED.

5.5.3.7.1 (Overview) says when a reservation key of a reservation holder has been removed by PREEMPT or PREEMPT AND ABORT and the released persistent reservation was a Registrants Only type, the device server shall establish a unit attention for "all registered initiators other than the initiator that issued the PR OUT" and set the additional sense code to RESERVATIONS RELEASED.

5.5.3.7.3.3 (PREEMPT) says that, when preempting a persistent reservation, REGISTRATIONS PREEMPTED is returned "for any initiator that lost its persistent reservation and/or registration." 5.5.3.7.3.4 (PREEMPT AND ABORT) inherits this behavior.

These conflict with each other and don't cover all the possible cases.

Example:
One set of initiators is registered with key of A.
Another set of initiators is registered with key of B.
One of the initiators (Z) with a key of A holds a Registrants Only persistent reservation.
All the registered initiators have permission to use the logical unit.
A valid PREEMPT (with a Service Action Key of A or B) is run from a registered initiator…

SERVICE ACTION RESERVATION KEY = A **These involve losing the reservation:**	**Proposed actions**
initiator Z preempts A	Z loses old reservation Z maintains its registration Z owns new reservation (maybe with a different type/scope) all the As except Z lose registration and get REGISTRATIONS PREEMPTED if the reservation changed type at all, all the Bs get RESERVATIONS RELEASED (and see issue 2)
initiator Y (not Z) with a key of A preempts A	Z loses old reservation Y maintains registration all the As including Z except Y lose registration and get REGISTRATIONS PREEMPTED Y owns new reservation (maybe with a different type/scope) if the reservation changed type at all, all the Bs get RESERVATIONS RELEASED
initiator W with a key of B preempts A	Z loses old reservation all the As lose registration and get REGISTRATIONS PREEMPTED W owns the reservation if the reservation changed type at all, all the Bs get RESERVATIONS RELEASED
SERVICE ACTION RESERVATION KEY = B **These do not involve losing the reservation:**	
initiator Z preempts B	all the Bs lose registration all the Bs get REGISTRATIONS PREEMPTED
initiator Y (not Z) with a key of A preempts B	all the Bs lose registration all the Bs get REGISTRATIONS PREEMPTED
initiator W with a key of B preempts B	W keeps registration all the other Bs lose registration all the other Bs get REGISTRATIONS PREEMPTED [rev 1: RELEASED was a typo]

Issue #1a: Which additional sense code for other reasons
The overview section does not mention that unit attentions are also created when reservations are released by other means - CLEAR, REGISTER, and REGISTER AND IGNORE EXISTING KEY. They should all be described together.

5.5.3.7.5 (CLEAR) says that, when a key is removed due to a CLEAR, RESERVATIONS PREEMPTED is returned (for all registered initiators).

5.5.3.7.1 (Overview) says that a key may be removed (implying a reservation may be released) due to a REGISTER or REGISTER AND IGNORE EXISTING KEY with key=0 from the reservation holder, but does not indicate which additional sense code is returned. It should return

RESERVATIONS RELEASED. Also, this is listed in the removal list but not the release list (it causes both to happen).

5.5.3.4 (Registering) discusses the case of a nonzero key (how to register) but doesn't mention what happens when a key of zero is used (removes registration and possibly releases a reservation). It should at least point to the release section of text.

Issue #2: Preempting self

5.5.3.7.3.3 (PREEMPT) says that preempting with the new key (SERVICE ACTION RESERVATION KEY) equal to the old key (RESERVATION KEY) is not an error; the reservation key (registration) remains in place and a new persistent reservation is established. However, the list of preempt actions direct the device server to "remove the registration for the initiators specified" which would include the initiator running the PREEMPT. Its key needs to remain for the new persistent reservation to be established.

Issue #3: Aborting self

5.5.3.7.3.4 (PREEMPT AND ABORT) needs to be clear that a PREEMPT AND ABORT does not abort the PERSISTENT RESERVE OUT command itself.

Issue #4: Linked command conflicts

5.5.1 (overview) describes when commands are checked for reservation conflicts and refers to the Enabled task state. The description is not accurate for linked commands. Tasks enter task states, not commands. Tasks may consist of more than one command. Found during discussion with Mallikarjun Chadalapaka (HP).

Issue #4b: What if one of the linked commands is a PERSISTENT RESERVE OUT which changes the reservation state? Add a rule that no commands in a linked set may follow a command that changes the reservation state.

Issue #5: Problems incorporating 01-099 All target ports

The description of what is saved for a reservation has a line from the registration data list, not the reservation list.

Issue #6: Problems with 01-204 All Registrants type

A Unit Attention is not generated for All Registrants when the last registration is removed, since there are no registered initiators left to notify. However, a UA should be generated if a PREEMPT changes the TYPE or SCOPE while registered initiators are present.

PREEMPT behavior under this new type is not described at all. Issues include:
a) what does PREEMPT of key of zero mean? It could mean release the reservation (since it is reported as a key of zero in READ RESERVATIONS). Or it could be disallowed.
b) what does PREEMPT of key of non-zero mean? With the Registrants-Only type, this splits into cases where the key is that of the reservation holder (preempt the reservation and remove the registration) and where it is not (remove the registration). Either has merit for All Registrants.

The goal of All Registrants is to let initiators de-register without losing the reservation. However, 01-204 did not state which service actions should de-register only - PREEMPT? REGISTER with service action key of 0?

This proposal suggests that PREEMPT of zero changes the reservation. PREEMPT of a non-zero key removes the registration but does not change the reservation. REGISTER of zero removes the registration but does not change the reservation. RELEASE releases the reservation.

Example:
One set of initiators (including Z) is registered with key of A.
Another set of initiators (including Y) is registered with key of B.
An All Registrants persistent reservation is in effect.
All the registered initiators have permission to use the logical unit.
A valid PR OUT is run from a registered initiator …

Service Action (always from a registered initiator)	**Proposed actions**
initiator runs CLEAR	remove all registrations release reservation RESERVATIONS PREEMPTED to all initiators
initiator runs PREEMPT (0)	don't change any registrations release reservation establish new reservation if the scope or type changed, RESERVATIONS RELEASED
initiator Z runs PREEMPT (A)	all the As except Z lose registration REGISTRATIONS PREEMPTED to all As except Z
initiator Y runs PREEMPT (A)	all the As lose registration REGISTRATIONS PREEMPTED to all As
initiator Z runs PREEMPT (B)	all the Bs lose registration REGISTRATIONS PREEMPTED to all Bs
initiator Y runs PREEMPT (B)	all the Bs except Y lose registration REGISTRATIONS PREEMPTED to all Bs except Y
initiator runs RELEASE	release reservation RESERVATIONS RELEASED to all initiators
initiator Z runs REGISTER with SERVICE ACTION RESERVATION KEY of zero	remove registration from Z
initiator Y runs REGISTER with SERVICE ACTION RESERVATION KEY of zero	remove registration from Y

Suggested Changes to SPC-3
3.1.63 persistent reservation holder: The initiator port or ports that are allowed to release or change a persistent reservation without preempting it (see 5.5.3.6).

5.5 Reservations

5.5.1 Reservations overview

Reservations may be used to allow a device server to execute commands from a selected set of initiator ports. The device server shall and reject commands from initiator ports outside the selected set of initiator ports. The device server by uniquely identifies ying initiator ports using protocol specific mechanisms.
[Editor's note: still a global problem with "initiator port" usage since most reservation rules apply to the I_T, not just the I. This issue is generally ignored by this proposal.]

Application clients may add or remove initiator ports 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.
…
Reservations may be further qualified by restrictions on types of access (e.g., read, write, control). However, any restrictions based on the type of reservation are independent of the scope of the reservation. In addition, some methods of reservation management permit establishing reservations on behalf of another device initiator port in the same SCSI domain (third-party reservations).

…
In table 10 and table 11 the following key words are used:
allowed: Commands issued by initiators not holding the reservation or by initiators not registered when a registrants only persistent reservation is present should complete normally.
conflict: Commands issued by initiators not holding the reservation or by initiators not registered when a registrants only persistent reservation is present shall not be performed and the device server shall terminate the command with a RESERVATION CONFLICT status.

Commands from initiators holding a reservation should complete normally. The behavior of commands from registered initiators when a registrants only persistent reservation is present is specified in table 10 and table 11. The all registrants persistent reservation types are included as registrants only types.

[Editor's note: issue #4 fixed below]
~~A command that does not explicitly write the medium shall be checked for reservation conflicts before the command enters the enabled task state for the first time. Once the command has entered the enabled task state, it shall not be terminated with a RESERVATION CONFLICT due to a subsequent reservation.~~

~~A command that explicitly writes the medium shall be checked for reservation conflicts before the device server modifies the medium or cache as a result of the command. Once the command has modified the medium, it shall not be terminated with a RESERVATION CONFLICT due to a subsequent reservation.~~
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 a related command standard (see 3.1.15) 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 10 - SPC commands that are allowed in the presence of various reservations
…
Table 11 - PERSISTENT RESERVE OUT service actions that are allowed
…
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 reserve/release command or persistent reserve service action shall be performed as a single indivisible event.

Multiple reserve/release commands or persistent reserve service actions may be present in the task set at the same time. The order of execution of such commands is defined by the tagged queu~~e~~ing restrictions, if any, but each is executed as a single indivisible command without any interleaving of actions that may be required by other reservation commands.

5.5.2 The Reserve/Release management method
The reserve/release management method commands, RESERVE(6), RESERVE(10), RELEASE(6), and RELEASE(10) are used among multiple initiator ports that do not require operations to be protected across initiator failures (and subsequent hard resets). The reserve/release reservations management method also allows an application client to provide restricted device access to one additional initiator port (a third-party initiator port), usually a

temporary initiator performing a service for the application client sending the reservation command.

Reservations managed using the reserve/release method do not persist across some recovery actions (e.g., hard resets). When a ~~target~~ logical unit performs one of these recovery actions, the application client(s) have to rediscover the configuration and re-establish the required reservations. Reserve/release managed reservations are retained by the device server until released or until reset by mechanisms specified in this standard.

The RESERVE(6) and RESERVE(10) commands allow superseding reservations.

5.5.3 The Persistent Reservations management method

5.5.3.1 Overview of the Persistent Reservations management method

…

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 multiple target ports. Before a persistent reservation may be established, an initiator port shall register with a device server using a reservation key. Reservation keys are necessary to allow:

…

[Editor's note: (editorial) same key different T is also allowed but that's not what this is trying to describe. "same key for the same T_L" might work.]

Reservation key values may be used by application clients to identify initiator ports, using application specific methods that are outside the scope of this standard. This standard provides the ability to register no more than one key per I_T_L nexus. Multiple initiator ports may use the same key for a logical unit accessed through the same target port. An initiator port may establish registrations for multiple logical units in a SCSI target device using any combination of unique or duplicate keys. These rules provide the ability for an application client to preempt multiple initiator ports with a single PERSISTENT RESERVE OUT command, but they do not provide the ability for the application client to uniquely identify the initiator ports using the PERSISTENT RESERVE commands.

5.5.3.2 Preserving persistent reservations and reservation keys

…

[Editor's note: T10 reflector traffic indicated that item a), added per the January WG, is confusing and the March WG agreed it should be removed.]

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

~~a) Indication of whether the registration information is valid;~~

b) On SCSI protocols where initiator port names (see 3.1.41) are required, the initiator port name; otherwise, the initiator port identifier (see 3.1.40);

c) Reservation key; and

d) Indication of the target port to which the registration was applied.

[Editor's note: issue #5: the reservation info includes the T used in the reservation, not the registration. The registration info text (not shown) properly includes the T from the registration.]

The device server shall preserve the following information about the existing persistent reservation ~~information~~ across any reset, and if the persist through power loss capability is enabled, across any power cycle:

~~a) Indication of whether the reservation information is valid;~~

b) On SCSI protocols where initiator port names are required, the initiator port name; otherwise, the initiator port identifier;

c) Reservation key;

d) Scope;

e) Type; and

f) Indication of ~~the~~ which target port ~~to which the registration was applied~~through which the reservation was established.

For an All Registrants type persistent reservation, only the scope and type need to be preserved.

…

5.5.3.4 Registering

To establish a persistent reservation the initiator shall first register an initiator port with a logical unit. An initiator registers with a logical unit by issuing a PERSISTENT RESERVE OUT command with REGISTER or REGISTER AND IGNORE EXISTING KEY service action.

If the initiator port has not yet established a reservation key or the reservation key and registration have been removed, the registration is accomplished by issuing a PERSISTENT RESERVE OUT command with REGISTER service action 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.

[Editor's note: issue #1a: non-zero behavior is described but this section is silent on zero behavior.]

If the initiator port has an established registration it may change its reservation key. This is accomplished by issuing a PERSISTENT RESERVE OUT command with REGISTER service action with the following parameters:

a) APTPL bit optionally set to one;
[Editor's note: rev 1: use nexus wording]
b) RESERVATION KEY field set to the value of the ~~previously established~~I_T_L nexus' registered reservation key; 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 5.5.3.7.1).

Alternatively, an initiator may establish a reservation key for an initiator port without regard for whether one has previously been established by issuing a PERSISTENT RESERVE OUT command with REGISTER AND IGNORE EXISTING KEY service action 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, the registration shall be removed (see 5.5.3.7.1).

…

It is not an error for an initiator port that is registered to register 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 initiator port is registered with the same reservation key and one of those initiator ports registers again it has no effect on the other initiator port's 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 initiator port may be changed without affecting any previously created persistent reservation).

Multiple initiator ports may register with the same reservation key. A ~~SCSI initiator device~~n initiator port may use the same reservation key for ~~multiple~~ other logical units.

5.5.3.5 ~~Creating a persistent reservation when there is no persistent reservation~~Reserving

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

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

Only one persistent reservation with a scope of logical unit is allowed at a time per logical unit. Multiple persistent reservations with a scope of element may be created in a logical unit that contains multiple elements. However, there shall only be one persistent reservation per element.

If the device server receives a PERSISTENT RESERVE OUT command from an initiator port other than a persistent reservation holder (see 5.5.3.6) 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.

If the target receives a RESERVE(10) or RESERVE(6) command when a persistent reservation exists for the logical unit then the command shall be rejected with a RESERVATION CONFLICT.

See 5.5.1 for information on when a persistent reservation takes effect.

[Editor's note: add "persistent" to avoid confusion with reserve/release reservations]

5.5.3.6 The persistent reservation holder

A persistent reservation holder has its reservation key returned in the parameter data from a PERSISTENT RESERVE IN command with READ RESERVATIONS service action.

The persistent reservation holder is determined by the type of 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 initiator port as indicated by a reservation key of zero; or

b) For all other persistent reservation types, the persistent reservation holder is the initiator port from which the PERSISTENT RESERVE OUT command with RESERVE, PREEMPT, or PREEMPT AND ABORT service action was received as indicated by the reservation key of that initiator port.

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 5.5.3.5).

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 5.5.3.7.1), the persistent reservation is automatically released. When the persistent reservation holder is more than one initiator port, the persistent reservation is not automatically released until the registrations for all persistent reservation holder initiator ports are removed.

5.5.3.7 Releasing persistent reservations and removing registrations

5.5.3.7.1 Overview of releasing persistent reservations and removing registrations

[Editor's note: (editorial) specifying the field names makes this section clearer since they're so similar.]

An application client may ***release or preempt ~~a~~ persistent reservations*** by issuing one of the following commands through a registered initiator port with the RESERVATION KEY field set to ~~using~~ the value of the I_T_L nexus' registered reservation key:

a) A PERSISTENT RESERVE OUT command with RELEASE service action from ~~the~~ a persistent reservation holder (see 5.5.3.7.2);

b) A PERSISTENT RESERVE OUT command with PREEMPT service action specifying the reservation key of the persistent reservation holder(s) ~~the persistent reservation~~ (see 5.5.3.7.3);
c) A PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action specifying the reservation key of the persistent reservation holder(s) ~~the persistent reservation~~ (see 5.5.3.7.4); ~~or~~
d) A PERSISTENT RESERVE OUT command with CLEAR service action (see 5.5.3.7.5); ~~.~~ or
e) A PERSISTENT RESERVE OUT command with REGISTER service action or REGISTER AND IGNORE EXISTING KEY service action with the SERVICE ACTION RESERVATION KEY field set to zero, if the initiator port is the persistent reservation holder and the persistent reservation is not an All Registrants type (see 5.5.3.7.x).

An application client may ***remove*** ~~***a registration***~~***s*** by issuing one of the following commands through a registered initiator port with the RESERVATION KEY field set to ~~using~~ the value of the I_T_L nexus' registered reservation key:
a) A PERSISTENT RESERVE OUT command with PREEMPT service action ~~specifying~~ with the SERVICE ACTION RESERVATION KEY field set to ~~the~~ the reservation key (see 5.5.3.7.3) to be removed;
b) A PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action with the SERVICE ACTION RESERVATION KEY field set to ~~specifying~~ the reservation key (see 5.5.3.7.4) to be removed;
c) A PERSISTENT RESERVE OUT command with CLEAR service action (see 5.5.3.7.5); or
d) A PERSISTENT RESERVE OUT command with REGISTER service action or ~~a~~ REGISTER AND IGNORE EXISTING KEY service action with ~~zero in~~ the [small caps service action] SERVICE ACTION RESERVATION KEY field set to zero (see 5.5.3.7.x)~~, if the initiator port is the same as the initiator port that registered the reservation key~~.
[Editor's note: it cannot run the command if the RESERVATION KEY was not correct. The if statement is confusing.]

When a reservation key (i.e, registration) has been removed, no information shall be reported for that unregistered initiator port in subsequent READ KEYS service action(s) until the initiator port is registered again (see 5.5.3.4).

The handling of any persistent reservation whose persistent reservation holder ~~or holder~~(s) become unregistered shall be as described in table 12.
[Editor's note: issues #1 and #6 addressed in this table.]

Reservation type	~~**Handling persistent reservation that results from an initiator port unregistering**~~**When to release a persistent reservation and generate unit attention conditions**	~~**Release Unit Attention Required**~~
Write Exclusive – Registrants Only or Exclusive Access – Registrants Only	When the persistent reservation holder (see 5.5.3.6) becomes unregistered the persistent reservation shall be released. ~~The device server shall establish a unit attention condition for all registered initiator ports other than the initiator port that unregistered The sense key shall be set to UNIT ATTENTION and the additional sense data shall be set to RESERVATIONS RELEASED.~~ The device server shall establish a unit attention for each registered initiator port whose reservation key was removed. The additional sense code shall be set as follows: a) If the service action was CLEAR, the additional sense code shall be set to RESERVATIONS PREEMPTED; or b) If the service action was PREEMPT or PREEMPT AND ABORT, the additional sense code shall be set to REGISTRATIONS PREEMPTED.	~~yes~~

	If the TYPE or SCOPE changed or the reservation was released, the device server shall establish a unit attention for each registered initiator port whose reservation key was not removed. The additional sense code shall be set as follows: a) If the service action was PREEMPT or PREEMPT AND ABORT, the additional sense code shall be set to RESERVATIONS RELEASED; b) If the service action was REGISTER or REGISTER AND IGNORE with the SERVICE ACTION KEY field set to zero, the additional sense code shall be set to RESERVATIONS RELEASED; or c) If the service action was RELEASE, the additional sense code shall be set to RESERVATIONS RELEASED.	
Write Exclusive – All Registrants or Exclusive Access – All Registrants	The persistent reservation shall be released ~~only~~ when the registration for the last registered initiator port ~~unregisters~~is removed or when the TYPE or SCOPE is changed. The device server shall establish a unit attention for each registered initiator port whose reservation key was removed. The additional sense code shall be set as follows: a) If the service action was CLEAR, the additional sense code shall be set to RESERVATIONS PREEMPTED; b) If the service action was PREEMPT or PREEMPT AND ABORT with a SERVICE ACTION RESERVATION KEY set to nonzero, the additional sense code shall be set to REGISTRATIONS PREEMPTED. If a persistent reservation was removed or changed, the device server shall establish a unit attention for each registered initiator port whose reservation key was not removed. The additional sense code shall be set as follows: a) If the service action was PREEMPT or PREEMPT AND ABORT with a SERVICE ACTION RESERVATION KEY set to zero, the additional sense code shall be set to RESERVATIONS RELEASED. b) If the service action was RELEASE, the additional sense code shall be set to RESERVATIONS RELEASED.	~~No for unregister Yes for release~~
All other reservation types	When the persistent reservation holder (see 5.5.3.6) becomes unregistered the persistent reservation shall be released.	~~No~~

[Editor's note: Remove the "Release Unit Attention Required" column.]

[Editor's note: issue #6: All Registrants should be notified if one of them changes the type or scope with a PREEMPT.]
[Editor's note: deleting this table in favor of the subsections' text should be considered.]

~~A~~ Registrations and persistent reservations may also be released by a loss of power, if the persist through power loss capability is not enabled. When the most recent APTPL value received by the device server is zero (see 7.12.3), a power cycle:
a) Performs a hard reset;
b) Releases all persistent reservations; and
c) Removes all registered reservation keys (see 5.5.3.4).

5.5.3.7.2 Releasing ~~a persistent reservation~~
Only ~~the~~ a persistent reservation holder (see 5.5.3.6) is allowed to release ~~that~~ a persistent reservation.

An application client releases a persistent reservation held by an initiator port by issuing a PERSISTENT RESERVE OUT command with RELEASE service action through a ~~registered initiator port that is a~~persistent reservation holder with the following parameters:
a) RESERVATION KEY field set to the value of the I_T_L nexus' registered reservation key; and
b) TYPE and SCOPE fields set to match the persistent reservation being released.

In response to a persistent reservation release request from a reservation holder the device server shall perform a release by doing the following as an uninterrupted series of actions:
a) ~~Releasing~~ Release the persistent reservation;
b) Not ~~removing~~ remove any registration(s);
c) If the released persistent reservation ~~has one of the types that table 12 lists as requiring a unit attention after a release (see 5.5.3.7.1)~~is of a Registrants Only or All Registrants type, the device server shall establish a unit attention condition for all registered initiator ports other than the initiator port that issued the PERSISTENT RESERVE OUT command with RELEASE service action. The ~~sense key shall be set to UNIT ATTENTION and the~~ additional sense ~~data~~ code shall be set to RESERVATIONS RELEASED; and
d) If the persistent reservation is of any other type the device server shall not establish a unit attention condition.

The established persistent reservation shall not be altered and the device server shall return a CHECK CONDITION status for a PERSISTENT RESERVE OUT command that specifies the release of a persistent reservation when:
a) The requesting initiator port is a persistent reservation holder (see 5.5.3.6); and
b) The SCOPE and TYPE fields do not match the scope and type of the established persistent reservation.

The sense key shall be set to ILLEGAL REQUEST and additional sense data shall be set to INVALID RELEASE OF PERSISTENT RESERVATION.

If there is no persistent reservation, or in response to a persistent reservation release request from a registered initiator port ~~using a reservation key value that does not match that of the reservation holder~~that is not a persistent reservation holder, the device server shall do the following:
a) Not release the persistent reservation (if any);
b) Not remove ~~or change~~ any registration(s); and
c) Return a status of GOOD.

[Editor's note: (editorial) there was no separate subsection 5.5.3.7.x for using REGISTER or REGISTER AND IGNORE EXISTING KEY with a key of zero. Use "unregistering" to describe this.]
5.5.3.7.x Unregistering
An application client may remove a registration for an initiator port by issuing a PERSISTENT RESERVE OUT command with REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action with the SERVICE ACTION RESERVATION KEY field set to zero through that initiator port.

If the initiator port is a reservation holder, the persistent reservation is of an All Registrants type, and the initiator port was the last registered initiator, the device server shall also release the persistent reservation.

If the initiator port is the reservation holder and the persistent reservation is of a type other than All Registrants, the device server shall also release the persistent reservation. If the persistent reservation is a Registrants Only type, the device server shall generate a unit attention condition for all registered initiators. The additional sense code shall be set to RESERVATIONS RELEASED.

[Editor's note: (editorial) Preempting might only affect registrations, so take persistent reservation out of the titles]

5.5.3.7.3 Preempting ~~an existing persistent reservation~~

5.5.3.7.3.1 Overview of preempting ~~an existing persistent reservation~~

[Editor's note: (editorial) this section affects both PREEMPT and PREEMPT AND ABORT, so mention both everywhere.]

A PERSISTENT RESERVE OUT command with PREEMPT service action or PREEMPT AND ABORT service action is used to either preempt (i.e., replace) a persistent reservation and remove a registration, ~~/~~or just remove a registration. Table xx lists the actions taken based on the current persistent reservation type and the SERVICE ACTION RESERVATION KEY. ~~The determination of whether the preempt relates to a persistent reservation or a registration is made by the device server by examining the value in the SERVICE ACTION RESERVATION KEY field of the PREEMPT service action.~~

~~If the value in the SERVICE ACTION RESERVATION KEY field is the reservation holder of the persistent reservation being preempted then the persistent reservation is preempted and any matching registration(s) removed (see 5.5.3.7.3.3).~~

~~If the value in the SERVICE ACTION RESERVATION KEY field is not the reservation holder of the persistent reservation, the persistent reservation shall not be preempted but any matching registration(s) shall be removed (see 5.5.3.7.3.4).~~

Table xx. PREEMPT or PREEMPT AND ABORT action

Persistent reservation type	**SERVICE ACTION RESERVATION KEY**	**Action**	**Reference**
All Registrants	Zero	Preempt persistent reservations and remove registrations	5.5.3.7.3.3
	Any reservation key	Remove registrations	5.5.3.7.3.4
All other types	Zero	Return CHECK CONDITION status, setting the sense key to ILLEGAL REQUEST and the additional sense code to INVALID FIELD IN PARAMETER LIST	
	The reservation holder's reservation key	Preempt persistent reservations and remove registrations	5.5.3.7.3.3
	Any other reservation key	Remove registrations	5.5.3.7.3.4

See figure 2 for a description of how a device server interprets a PREEMPT service action to determine its actions (e.g., preempt persistent reservation, remove registration, or both preempt persistent reservation and remove registration).

…

Diamond: SERVICE ACTION RESERVATION KEY matches reservation key of any persistent reservation holder?

…

Figure 2 - Device server interpretation of PREEMPT service action

[Editor's note: (technical) the figure needs to incorporate zero handling. Left to the editor.]
…

5.5.3.7.3.2 Failed persistent reservation preempt

[Editor's note: (technical) a single BUSY or TASK SET FULL should not result in a LOGICAL UNIT RESET.]

If the preempting initiator port's PREEMPT service action or PREEMPT AND ABORT service action fails (e.g., repeated TASK SET FULL status, repeated BUSY status, ~~SCSI~~ protocol time-out, or time-out due to the queue being blocked due to a failed initiator port), the application client may issue a LOGICAL UNIT RESET task management function to the failing logical unit to remove blocking tasks and then reissue the preempting service action.

5.5.3.7.3.3 Preempting persistent reservations and removing registrations

~~Any registered initiator port~~An application client may preempt any 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 initiator port with the following parameters:

a) RESERVATION KEY field set to the value of the I_T_L nexus' registered reservation key;
b) SERVICE ACTION RESERVATION KEY set to ~~match~~ the value of the reservation key of the persistent reservation ~~being~~ to be preempted; and
c) TYPE and SCOPE fields set to define a new persistent reservation. The SCOPE [add small caps to scope] and TYPE of the persistent reservation created by the preempting initiator port may be different than those of the persistent reservation being preempted.

If the SERVICE ACTION RESERVATION KEY identifies ~~the~~ a persistent reservation holder ~~a reservation~~ (see 5.5.3.6), 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 ~~specified in the PERSISTENT RESERVE OUT parameter list~~;

[Editor's note: issue #2 fixed below]

b) Remove the registration(s) for the initiator port or initiator ports identified by the SERVICE ACTION RESERVATION KEY ~~specified in the PERSISTENT RESERVE OUT parameter list (see 5.5.3.4)~~ except for the initiator port that issued the PERSISTENT RESERVE OUT command;
c) Establish a persistent reservation for the preempting initiator port using the contents of the SCOPE and TYPE fields;
d) Process tasks as defined in 5.5.1; and
e) Establish a unit attention condition for ~~any~~ each initiator port that lost its persistent reservation and/or registration. The sense key shall be set to UNIT ATTENTION and the additional sense data 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 initiator port.

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 initiator port:

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.

[Editor's note: (editorial) Either remove the service action or list both below. This paragraph doesn't refer to "preempt" so repeating the service actions seems helpful.]
A PERSISTENT RESERVE OUT ~~specifying a~~with PREEMPT service action or PREEMPT AND ABORT service action with the SERVICE ACTION RESERVATION KEY value equal to the reservation holder's reservation key [Editor's note: rev 1: reworded "reservation key" to better indicate the key of the current reservation, not the RESERVATION KEY field] is not an error. In that case the device server shall establish the new persistent reservation and maintain the registration.

5.5.3.7.3.4 Removing registrations

When a registered reservation key ~~is not associated with a persistent reservation~~does not identify a persistent reservation holder, an application client may remove the registration(s) without affecting any persistent reservations by issuing a PERSISTENT RESERVE OUT command with PREEMPT service action through a registered initiator port with the following parameters:

a) RESERVATION KEY field set to the value of the I_T_L nexus' registered reservation key; and
b) SERVICE ACTION RESERVATION KEY field set to match the reservation key of the registration being removed.

If the SERVICE ACTION RESERVATION KEY field ~~is not associated with a~~ does not identify a persistent reservation holder the device server shall perform a preempt by doing the following in an uninterrupted series of actions:

a) Remove the registration for the initiator port or initiator ports identified by the SERVICE ACTION RESERVATION KEY field ~~specified in the PERSISTENT RESERVE OUT parameter list~~;
b) Ignore the contents of the SCOPE and TYPE fields;
c) Process tasks as defined in 5.5.1; and
d) Establish a unit attention condition for any initiator port that lost its registration other than the initiator port that sent the PERSISTENT RESERVE OUT command ~~with PREEMPT service action~~. The sense key shall be set to UNIT ATTENTION and the additional sense data shall be set to REGISTRATIONS PREEMPTED.

If a PERSISTENT RESERVE OUT ~~specifying a~~with PREEMPT service action or PREEMPT AND ABORT service action sets the SERVICE ACTION RESERVATION KEY field to a value that does not match any registered reservation key the device server shall return a RESERVATION CONFLICT status.

5.5.3.7.4 Preempting ~~an existing persistent reservation~~ and ~~concurrently~~ aborting ~~tasks~~

The initiator port'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 5.5.3.7.3) except for the following additions. If no reservation conflict occurred, the device server shall perform the following uninterrupted series of actions:

[Editor's note: (editorial) item c) up to the top. Describe the case where the PR OUT is not processed before describing the cases where it is processed.]

a) If the TST field is 000b (see 8.4.6) and ACA or CA conditions exist for initiator ports other than the initiator port 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 equals zero. If the TST field contains 001b, then ACA or CA conditions for initiator ports other than the initiator port being preempted shall not prevent the processing of the PERSISTENT RESERVE OUT command;
b~~a~~) Perform the uninterrupted series of actions described for the PREEMPT service action (see 5.5.3.7.3);

[Editor's note: issue #3 fixed below]

c~~b~~) All tasks from preempted initiator ports (called preempted tasks) except for the task containing the PERSISTENT RESERVE OUT command itself shall be terminated ~~and~~. A~~a~~pplication client notification shall be provided, as specified by the TAS bit in the Control mode page (see 8.4.6) that applies to 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). ~~The~~ Any preempted tasks from the initiator port that sent the PREEMPT AND ABORT service action ~~(if any)~~ 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~~e) The device server shall clear any ACA or CA condition associated with an initiator port being preempted and shall clear any tasks with an ACA attribute from that initiator port. ~~If the TST field is 000b (see 8.4.6) and ACA or CA conditions exist for initiator ports other than the initiator port 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 equals zero. If the TST field contains 001b, then ACA or CA conditions for initiator ports other than the initiator port being preempted shall not prevent the processing of the PERSISTENT RESERVE OUT command~~; and

~~e~~d) For ~~SCSI devices~~ logical units that implement the PREVENT ALLOW MEDIUM REMOVAL command, the device server shall perform an action equivalent to the execution of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field equal to zero for the initiator port or initiator ports being preempted (see 7.13).

The actions described in the preceding list shall be performed for all initiator ports that are registered with the SERVICE ACTION RESERVATION KEY value, without regard for whether the preempted initiator port(s) hold the persistent reservation.

Any asynchronous event reporting operations in progress are not affected by the PREEMPT AND ABORT service action.

5.5.3.7.5 Clearing ~~a persistent reservation~~

[Editor's note: (editorial) device server is inside a logical unit so no need to reference it]

Any application client may release a persistent reservation and remove all registrations from a device server ~~for a specific logical unit~~ by issuing a PERSISTENT RESERVE OUT command with CLEAR service action through a registered initiator port with the following parameter:

a) RESERVATION KEY field set to the value of the I_T_L nexus' registered reservation key.

In response to this request the device server shall perform a clear by doing the following as part of an uninterrupted series of actions:

a) Release any persistent reservation ~~associated with the logical unit~~;
b) Remove all registration(s) (see 5.5.3.4);
c) Ignore the contents of the SCOPE and TYPE fields;

[Editor's note: (editorial) the big reservation table uses "allowed", not "nonconflicting"]

d) Continue normal execution of any tasks from any initiator port that have been accepted by the device server as allowed (i.e., nonconflicting); and
e) Establish a unit attention condition for all registered initiator ports other than the initiator port that sent the PERSISTENT RESERVE OUT command with CLEAR service action, if any, for the cleared logical unit. The sense key shall be set to UNIT ATTENTION and the additional sense data shall be set to RESERVATIONS PREEMPTED.

Application clients should not use the CLEAR service action except during recoveries that are associated with initiator port or system reconfiguration, since the effect of the CLEAR service action is to remove the persistent reservation management conventions that protect data integrity.

[Editor's note: the PREEMPT section includes this advice, which should equally apply to CLEAR.] If the initiator port's CLEAR service action fails (e.g., repeated TASK SET FULL status, repeated BUSY status, protocol time-out, or time-out due to the queue being blocked due to a failed initiator port), the application client may issue a LOGICAL UNIT RESET task management function to the failing logical unit to remove blocking tasks and then reissue the CLEAR service action.

5.6 Multiple target port and initiator port behavior

SAM-2 specifies the behavior of logical units being accessed by application clients through more than one initiator port through one or more target ports. Additional initiator ports and target ports provide alternate paths through which the device server may be reached. An alternate path may be used to improve the availability of logical units in the presence of certain types of failures and to improve the performance between an application client and logical unit when some paths may be busy.

If a SCSI target device has more than one target port, the arbitration and connection management among the target ports is vendor specific. If one target port is being used by an initiator port, accesses attempted through other target port(s) may:

a) Receive a status of BUSY; or
b) Be accepted as if the other target port(s) were not in use.

The device server shall indicate the presence of multiple target ports by setting the MULTIP bit to one in its standard INQUIRY data.

[Editor's note: (editorial) all these should mention removing registrations, or none of them should.] [Editor's note: (editorial) "other" is only being defined for this a)-f) list. Remove the incomplete definition and let the references describe the details.]

~~For the purposes of handling reservations, other initiators are defined as all initiator ports on the same service delivery port except the initiator holding the reservation and all initiators on all other service delivery ports.~~ Only the following operations allow an initiator port to interact with the tasks of other initiator ports, regardless of the target port:

a) The PERSISTENT RESERVE OUT with PREEMPT service action preempts persistent reservations ~~for other initiator ports~~ (see 5.5.3.7.3);
b) The PERSISTENT RESERVE OUT with PREEMPT AND ABORT service action preempts persistent reservations and aborts ~~all~~ tasks ~~for other initiator ports~~ (see 5.5.3.7.4);
c) The PERSISTENT RESERVE OUT with CLEAR service action releases persistent reservations ~~and removes registrations~~ for all initiator ports (see 5.5.3.7.5);
d) The TARGET RESET task management function releases reservations established by the reserve/release method and removes all tasks for all initiator ports for all logical units accessible via the target port (see SAM-2). Persistent reservations remain unmodified;
e) The LOGICAL UNIT RESET task management function releases reservations established by the reserve/release method and removes all tasks for all initiator ports for the addressed logical unit and any logical units issuing from it in a hierarchical addressing structure (see SAM-2). Persistent reservations remain unmodified; and
f) The CLEAR TASK SET task management function removes all tasks for all initiator ports for the selected logical unit. Most other logical unit states remain unmodified, including MODE SELECT parameters, reservations, and ACA (see SAM-2).

7.11.4.3 Persistent Reservations Type

The value in the TYPE field shall specify the characteristics of the persistent reservation being established for all data blocks within the element or within the logical unit. Table 78 defines the characteristics of the different type values. For each persistent reservation type, table 78 lists

code value and describes the required device server support. In table 78, the description of required device server support is divided into three paragraphs:
1) A definition of the required handling for read operations;
2) A definition of the required handling for write operations; and
3) A definition of the persistent reservation holder (see 5.5.3.6).

Table 78 - Persistent reservation type codes

Write Exclusive Persistent r~~R~~eservation Holder: The initiator port that delivered the PERSISTENT RESERVE OUT command with RESERVE, PREEMPT, or PREEMPT AND ABORT service action as identified by its registered reservation key.
Exclusive Access Persistent r~~R~~eservation Holder: The initiator port that delivered the PERSISTENT RESERVE OUT command with RESERVE, PREEMPT, or PREEMPT AND ABORT service action as identified by its registered reservation key.
Write Exclusive - Registrants Only Persistent r~~R~~eservation Holder: The initiator port that delivered the PERSISTENT RESERVE OUT command with RESERVE, PREEMPT, or PREEMPT AND ABORT service action as identified by its registered reservation key.
Exclusive Access - Registrants Only Persistent r~~R~~eservation Holder: The initiator port that delivered the PERSISTENT RESERVE OUT command with RESERVE, PREEMPT, or PREEMPT AND ABORT service action as identified by its registered reservation key.
Write Exclusive - All Registrants Persistent r~~R~~eservation Holder: Any registered initiator port as identified by a zero reservation key value.
Exclusive Access - All Registrants Persistent r~~R~~eservation Holder: Any registered initiator port as identified by a zero reservation key value.

…
7.12.2 PERSISTENT RESERVE OUT Service Actions
…
The PERSISTENT RESERVE OUT command service actions are defined in table 86.

Table 86 — PERSISTENT RESERVE OUT service action codes

Code	Name	Description	PRGENERATION field incremented
00h	REGISTER	Register a reservation key with the device server (see 5.5.3.4) or unregister (see 5.5.3.7.x).	Yes
…			
04h	PREEMPT	Preempts persistent reservations and/or removes registrations~~from another initiator port~~ (see 5.5.3.7.3)	Yes
05h	PREEMPT AND ABORT	Preempts persistent reservations and/or removes registrations~~from another initiator port~~, and aborts all tasks for all initiator ports ~~registered with the specified reservation key~~preempted (see 5.5.3.7.3 and 5.5.3.7.4).	Yes
06h	REGISTER AND IGNORE EXISTING KEY	Register a reservation key with the device server (see 5.5.3.4) or unregister (see 5.5.3.7.x).	Yes

…
7.12.3 PERSISTENT RESERVE OUT parameter list
…
The SERVICE ACTION RESERVATION KEY field contains information needed for four service actions~~;~~: the REGISTER, REGISTER AND IGNORE EXISTING KEY, PREEMPT, and PREEMPT AND ABORT service actions. For the REGISTER and REGISTER AND IGNORE EXISTING KEY

service action, the SERVICE ACTION RESERVATION KEY field contains the new reservation key to be registered, or zero to unregister. For the PREEMPT and PREEMPT AND ABORT service actions, the SERVICE ACTION RESERVATION KEY field contains the reservation key of the persistent reservations that are being preempted and/or the reservation key to be removed. The SERVICE ACTION RESERVATION KEY field is ignored for all other service actions.