To: T10 Technical Committee
From: Rob Elliott, HP (elliott@hp.com)
Date: 31 October 2003
Subject: T10/03-337r2 SPC-3 Third party persistent reservations

Revision History

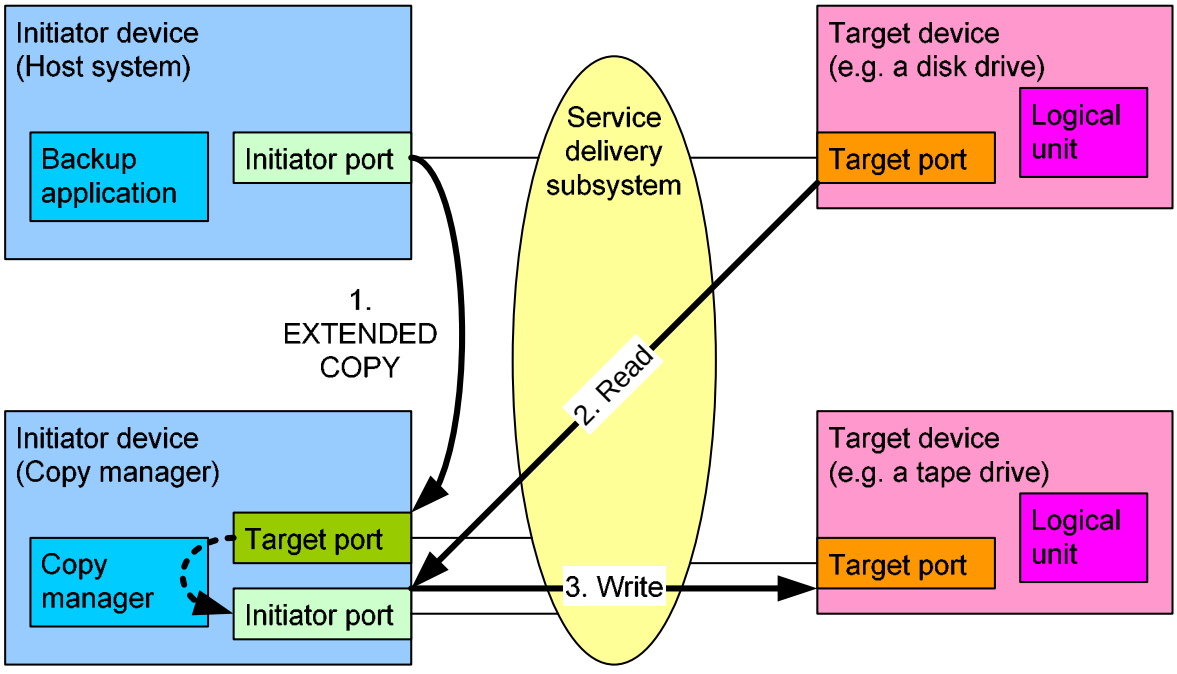
Revision 0 (3 October 2003) first revision
Revision 1 (16 October 2003) incorporated feedback from 10/14 conference call. Changed from adding a GIVE bit to REGISTER to a new service action called REGISTER AND MOVE. Changed from adding a TAKE bit to REGISTER to adding Specify Initiator Ports support to PREEMPT.
Revision 2 (31 October 2003) incorporated feedback from 10/28 conference call.

Related Documents

spc3r15 SCSI Primary Commands – 3 revision 15
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)

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.



Steps:
1. Backup app sends EXTENDED COPY to copy manager
2. Copy manager reads from one logical unit (e.g. a disk drive)
3. Copy manager writes to another (e.g. a tape drive)

Figure 1. Third-party copy

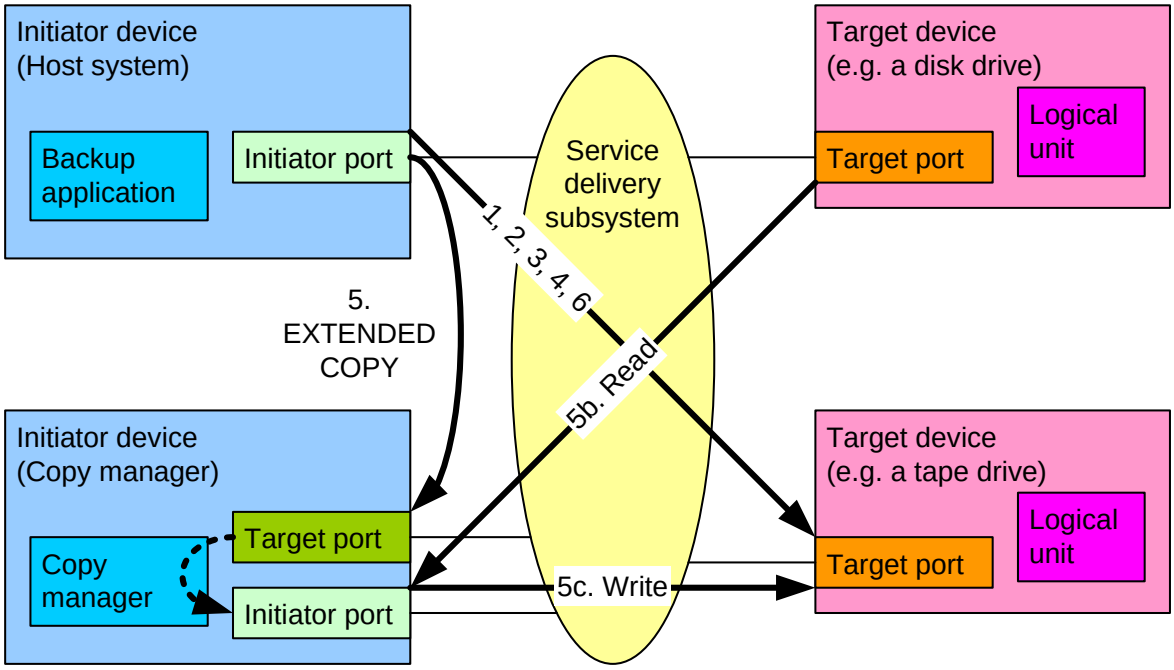
03-233 proposed a MOVE service action, specifying the reservation key to move. However, this isn't precise enough if more than one I_T is registered with the same reservation key. The Specify Initiator ports feature can be used to fix this.

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.

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.

Backup application steps
1. Register self with the destination (e.g. tape drive) with PR OUT/REGISTER
2. Reserve with PR OUT/RESERVE
3. Prepare destination (LOAD, REWIND, etc.)
4. Register copy manager and give reservation with PR OUT/REGISTER AND MOVE/ SPEC_I_PT
5. Send EXTENDED COPY to copy manager; wait for completion
6. Unregister copy manager and take back reservation with PR OUT/PREEMPT/SPEC_I_PT



Copy manager steps:
5a. Receive EXTENDED COPY command
5b. Copy manager reads from one logical unit (e.g. a disk drive)
5c. Copy manager writes to the reserved logical unit (e.g. a tape drive)
5d. Return status for EXTENDED COPY

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:

- SPEC_I_PT bit set to 1
- parameter data specifying the copy manager's initiator port (exactly one TransportID)

- SERVICE ACTION RESERVATION KEY set to a reservation key for the copy manager
- 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 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.

When it wants to revoke the handover, the backup application uses a PERSISTENT RESERVE OUT command PREEMPT or PREEMPT AND ABORT service action with:

- SPEC_I_PT bit set to 1
- parameter data specifying the copy manager's initiator port
- SERVICE ACTION RESERVATION KEY set to the copy manager's reservation key
- 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.

Suggested Changes

5.6 Reservations

5.6.1 Reservations overview

Reservations ~~may be~~are used to allow a device server to ~~execute~~ 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.

Reservations on one logical unit have no relationship with reservations on any other logical unit.

The scope of a reservation shall be one of the following:

a) **Logical unit reservations** - a logical unit reservation restricts access to the entire logical unit; and

b) **Element reservations** - an element reservation restricts access to a specified element within a medium changer.

Reservations ~~may be~~are further qualified by restrictions on types of access ~~(e.g., read, write).~~ :

a) Write Exclusive: reads shared, writes exclusive;

b) Exclusive Access: reads exclusive, writes exclusive;

c) Write Exclusive - Registrants Only: reads shared, writes exclusive;

d) Exclusive Access - Registrants Only: reads exclusive, writes exclusive;

e) Write Exclusive - All Registrants: reads shared, writes exclusive; and

f) Exclusive Access - All Registrants: reads exclusive, writes exclusive.

See table 102 in 6.11.4.3 for more information on types of reservations. ~~However, a~~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 31. If any element is reserved within a logical unit, that logical unit shall be considered reserved for the commands listed in table 31 and the allowed/conflict information in table 31 shall apply.

In table 31 and table 32 the following key words are used:

allowed: Commands received from I_T nexuses ~~not holding the reservation~~that are not persistent reservation holders (see 5.6.2.6) 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~~ that are not persistent reservation holders 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~~.~~ (see SAM-3 for prioritization of this status code versus other status codes like CHECK CONDITION).

Commands from I_T nexuses ~~holding a reservation~~ that are persistent reservation holders 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 31 and table 32.

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 3.1.17) 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 31 - SPC commands that are allowed in the presence of various reservations]

Table 32 — 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
(a) The reservation is not released (see 5.6.2.7.2).		

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 execution of such service actions is defined by the tagged queuing 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.

5.6.2 The Persistent Reservations management method

5.6.2.1 Overview of the Persistent Reservations management method

The persistent reservations management method is the mechanism specified by this standard for use by multiple ~~initiator ports~~ application clients communicating through multiple I_T nexuses that require operations to be protected 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 multiple target ports. Before a persistent reservation may be established, an application client shall register each I_T ~~N~~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~~es~~, if any, hold~~s~~ 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 ~~issuing~~ 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 key per I_T_L nexus. Multiple I_T nexuses may use the same key for a logical unit accessed through the same target port. Multiple initiator ports may use the same key value for a logical unit accessed through the same target ports. An initiator port may use the same key value for a logical unit accessed through different target ports.~~

A separate reservation key shall be maintained for each I_T nexus, regardless of the reservation key's value.

An I_T nexus may establish registrations for 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.

[the above paragraph seems to hint that a single PR OUT might affect more than one LU.]

The persistent reservation commands and service actions are summarized in table xx.

Table xx. Persistent reservations commands and service actions

Command	Service action	Description	Reference(s)
PERSISTENT RESERVE IN	READ KEYS	Report reservation keys	5.6.2.3.2, 6.11.2.2, and 6.11.3
	READ RESERVATION	Report persistent reservations	5.6.2.3.3, 6.11.2.3, and 6.11.4
	READ CAPABILITIES	Report capabilities	6.11.2.4 and 6.11.5
	READ FULL STATUS	*Report full status*	*5.6.2.3.x and 6.11.6*
PERSISTENT RESERVE OUT	REGISTER	Register Unregister	5.6.2.4 5.6.2.7.3
	REGISTER AND IGNORE EXISTING KEY	Register Unregister	5.6.2.4 5.6.2.7.3
	REGISTER AND MOVE	Register and move the reservation	5.6.2.x
	RESERVE	Reserve	5.6.2.5
	RELEASE	Release	5.6.2.7.2
	PREEMPT	Preempt	5.6.2.7.4
	PREEMPT AND ABORT	Preempt and abort	5.6.2.7.5
	CLEAR	Clear	5.6.2.7.6

[Editor's note: READ FULL STATUS is proposed in 03-342. That proposal also requests the 6.11.2 subsections be rearranged, which would change these references.]

5.6.2.2 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, ~~or~~ REGISTER AND IGNORE EXISTING KEY, or REGISTER AND MOVE service action.

After the application client enables the persist through power loss capability the device server shall preserve all current and future registrations and persistent reservations ~~associated with the logical unit to which the REGISTER or the REGISTER AND IGNORE EXISTING KEY 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 ~~or,~~ REGISTER AND IGNORE EXISTING KEY, 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) On SCSI protocols where initiator port names (see 3.1.46) are required, the initiator port name; otherwise, the initiator port identifier (see 3.1.45);

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) On SCSI 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 9 - 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~~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 above 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 165 (see 6.29).

5.6.2.3 Finding persistent reservations and reservation keys

5.6.2.3.1 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 PERSISTENT RESERVE IN commands with READ RESERVATION or READ KEYS service action.

5.6.2.3.2 Reporting reservation keys

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

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

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

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

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

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

5.6.2.3.3 Reporting persistent reservations

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

In response to a PERSISTENT RESERVE IN command with READ RESERVATION service action the device server shall report the following as an uninterrupted series of actions:

a) The current PRgeneration value (see 6.11.3);

b) The registered reservation key, if any, associated with the I_T nexus that holds the persistent reservation;

c) The scope and type of each persistent reservation, if any; and

d) The scope-specific address, if any (see 6.11.4.1).

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.

5.6.2.4 Registering

[Editor's note: this model section still assumes the I_T nexus used for PR OUT is the one being registered. With Specify Initiator Ports, that is no longer true. Since this proposal relies on Specify Initiator Ports, this section needs to be redone.]

~~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 or REGISTER AND IGNORE EXISTING KEY service action.~~

The PERSISTENT RESERVE OUT command REGISTER, REGISTER AND IGNORE EXISTING KEY, and REGISTER AND MOVE service actions are used to create, modify, or remove a registration for specified I_T nexus(es). The specified I_T nexus(es) are based on the SPEC_I_PT bit and ALL_TG_PT bits as described in table xx.

Table xx. SPEC_I_PT and ALL_TG_PT bits for REGISTER, REGISTER AND IGNORE EXISTING KEY, and REGISTER AND MOVE service actions

SPEC_I_PT bit	ALL_TG_PT bit	Specified I_T nexus(es)
0	0	The I_T nexus used for the PERSISTENT RESERVE OUT command
0	1	All I_T nexuses with: a) the same initiator port as that used for the PERSISTENT RESERVE OUT command; and b) all target ports
1	0	All I_T nexuses with: a) the initiator port(s) specified in additional parameter data; and b) the same target port as that used for the PERSISTENT RESERVE OUT command
1	1	All I_T nexuses with: a) the initiator port(s) specified in additional parameter data; and b) all target ports

Table xxx describes the REGISTER and REGISTER AND IGNORE EXISTING KEY service action behaviors based on the RESERVATION KEY field and the SERVICE ACTION RESERVATION KEY fields. The REGISTER AND MOVE service action behavior is described in 5.6.2.x.

Table xx. REGISTER and REGISTER AND IGNORE EXISTING KEY behaviors

I_T nexus used for the PERSISTENT RESERVE OUT command	Service action	RESERVATION KEY field	SERVICE ACTION RESERVATION KEY field	Result
Registered	REGISTER	other than the current reservation key	any	Return RESERVATION CONFLICT status
		the current reservation key	zero	Unregister (see 5.6.2.7.1)
		the current reservation key	non-zero	Change the reservation key(s) of the specified I_T nexus(es) to the value specified in the SERVICE ACTION RESERVATION KEY field
	REGISTER AND IGNORE EXISTING KEY	any	any	Return RESERVATION CONFLICT status
		any	zero	Unregister (see 5.6.2.7.1)
		any	non-zero	Change the reservation key(s) of the specified I_T nexus(es) to value specified in the SERVICE ACTION RESERVATION KEY field
Not registered	REGISTER	zero	zero	Return GOOD status
		zero	non-zero	Register the specified I_T nexus(es) with the value specified in the SERVICE ACTION RESERVATION KEY field
		non-zero	any	Return RESERVATION CONFLICT status
	REGISTER AND IGNORE EXISTING KEY	any	zero	Return GOOD status
		any	non-zero	Register the specified I_T nexus(es) with the value specified in the SERVICE ACTION RESERVATION KEY field

~~If the I_T nexus 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.~~

~~If the I_T nexus 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;~~

~~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 5.6.2.7.1).~~

~~Alternatively, an application client may establish a reservation key for an 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 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 5.6.2.7.1).~~

~~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 registration fails for any specified I_T nexus (e.g., if a specified I_T nexus is invalid, or if the SCSI target device does not have enough resources available to hold the registration information), none of the registrations shall be made.

If a PERSISTENT RESERVE OUT command with REGISTER ~~or a~~, REGISTER AND IGNORE EXISTING KEY, 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 REGISTER ~~or a~~, REGISTER AND IGNORE EXISTING KEY, 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 using the indication of the target port associated with the registration and either the initiator port name (see 3.1.46) on SCSI protocols where port names are required or the initiator port identifier (see 3.1.45) on SCSI 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 execute. ~~For each I_T nexus that is the source a PERSISTENT RESERVE OUT command with REGISTER 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 or the REGISTER AND IGNORE EXISTING KEY service action from the same I_T nexus or until the registration is removed (see 5.6.2.7).~~

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.

5.6.2.x Registering and moving the reservation

The PERSISTENT RESERVE OUT command REGISTER AND MOVE service action is used to register a specified I_T nexus and move a reservation from one I_T nexus to another.

The REGISTER AND MOVE service action shall be processed the same as the REGISTER service action (see 5.6.2.4), except as described in this subclause.

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; and

c) 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);

then the device server shall:

1) register the specified I_T nexus as described in 5.6.2.4;

2) 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.

NOTE: The I_T nexus that sent the PERSISTENT RESERVE OUT command remains registered. If the type of reservation is All Registrants, then 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 any of the aforementioned conditions is not met, the command shall be rejected with RESERVATION CONFLICT status.

5.6.2.5 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 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 I_T nexus other than a persistent reservation holder (see 5.6.2.6) 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.

[above "for the logical unit" is not true if element reservations are considered]

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.

5.6.2.6 Persistent reservation holder

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~~ all registered I_T nexus(es) are persistent reservation holders; or

b) For all other persistent reservation types, the persistent reservation holder is the I_T nexus that was granted the reservation from ~~which the~~ a PERSISTENT RESERVE OUT command with REGISTER AND MOVE, RESERVE, PREEMPT, or PREEMPT AND ABORT service action~~s was received~~.

A persistent reservation holder has its reservation key returned in the parameter data from a PERSISTENT RESERVE IN command with READ RESERVATION~~S~~ 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 5.6.2.5).

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 5.6.2.7.1), 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~~s I_T nexuses~~ are removed.

5.6.2.7 Releasing persistent reservations and removing registrations

5.6.2.7.1 Overview of releasing persistent reservations and removing registrations

An application client may release or preempt persistent reservations by issuing one of the following commands through a registered I_T nexus with the RESERVATION KEY field set to the reservation key value that is registered for the I_T ~~I~~ nexus:

a) A PERSISTENT RESERVE OUT command with RELEASE service action ~~from a persistent reservation holder~~ (see 5.6.2.7.2);

aa) If the persistent reservation is not an all registrants type and the I_T nexus is the persistent reservation holder, a PERSISTENT RESERVE OUT command with REGISTER or REGISTER AND IGNORE EXISTING KEY service action with the SERVICE ACTION RESERVATION KEY field set to zero (see 5.6.2.7.3);

b) A PERSISTENT RESERVE OUT command with PREEMPT service action ~~specifying the reservation key of the persistent reservation holder or holders~~ (see 5.6.2.7.4);

c) A PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action ~~specifying the reservation key of the persistent reservation holder or holders~~ (see 5.6.2.7.5); and

d) A PERSISTENT RESERVE OUT command with CLEAR service action (see 5.6.2.7.6).~~; or~~

~~e) If the I_T nexus is the persistent reservation holder and the persistent reservation is not an all registrants type, 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 (see 5.6.2.7.3).~~

An application client may remove registrations by issuing one of the following commands through a registered I_T nexus with the RESERVATION KEY field set to the reservation key value that is registered for the I_T ~~L~~ nexus:

aa) A PERSISTENT RESERVE OUT command with REGISTER or REGISTER AND IGNORE EXISTING KEY service action with the SERVICE ACTION RESERVATION KEY field set to zero (see 5.6.2.7.3);

a) A PERSISTENT RESERVE OUT command with PREEMPT service action ~~with the SERVICE ACTION RESERVATION KEY field set to the reservation key~~ (see 5.6.2.7.4) ~~to be removed~~;

b) A PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action ~~with the SERVICE ACTION RESERVATION KEY field set to the reservation key~~ (see 5.6.2.7.5) ~~to be removed~~;

c) A PERSISTENT RESERVE OUT command with CLEAR service action (see 5.6.2.7.6).~~;~~ ~~or~~

~~d) 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 (see 5.6.2.7.3).~~

When a ~~reservation key (i.e.,~~ registration~~)~~ has been removed, no information shall be reported for ~~that~~ an unregistered I_T nexus in subsequent READ KEYS service action(s) until the I_T nexus is registered again (see 5.6.2.4). As shown in table 33, the handling of any persistent reservation whose persistent reservation holder or holders become unregistered depends on the reservation type.

Table 33 — Handling for released persistent reservations

Reservation type	Reference
Write Exclusive – Registrants Only or Exclusive Access – Registrants Only	5.6.2.7.1.1
Write Exclusive – All Registrants or Exclusive Access – All Registrants	5.6.2.7.1.2
Write Exclusive or Exclusive Access	5.6.2.7.1.3

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 6.12.3), a power cycle:

a) Releases all persistent reservations; and

b) Removes all registered reservation keys (see 5.6.2.4).

[5.6.2.7.1.1 Handling for released registrants only persistent reservations]
[5.6.2.7.1.2 Handling for released all registrants persistent reservations]
[5.6.2.7.1.3 Handling for other released persistent reservations]
[5.6.2.7.2 Releasing]
5.6.2.7.3 Unregistering

An application client may remove a registration for an I_T nexus 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 I_T nexus.

If the I_T nexus is a reservation holder, the persistent reservation is of an all registrants type, and the I_T nexus was the last registered initiator, the device server shall also release the persistent reservation.

If the I_T nexus 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 every initiator port associated with a registered I_T nexus. The additional sense code shall be set to RESERVATIONS RELEASED.

5.6.2.7.4 Preempting

5.6.2.7.4.1 Overview of preempting

A PERSISTENT RESERVE OUT command with PREEMPT ~~service action~~ or PREEMPT AND ABORT service action is used to:

a) Preempt (i.e., replace) a persistent reservation and remove registrations; or

b) Remove registrations.

Table 34 lists the actions taken based on the current persistent reservation type and the SERVICE ACTION RESERVATION KEY field in the PERSISTENT RESERVE OUT command.

Table 34 — Preempting actions

Reservation type	Service action reservation key	Action	Reference
All registrants	Zero	Preempt persistent reservations and remove registrations.	5.6.2.7.4.3
	~~Any reservation key~~Non-zero	Remove registrations.	5.6.2.7.4.4
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.	
	Reservation holder's reservation key	Preempt persistent reservations and remove registrations.	5.6.2.7.4.3
	Any other ~~reservation key~~value	Remove registrations.	5.6.2.7.4.4

See figure 3 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).

[Figure 3]

Figure 3 — Device server interpretation of PREEMPT service action with SPEC_I_PT and ALL_TG_PT set to zero

The PERSISTENT RESERVE OUT command PREEMPT and PREEMPT AND ABORT service actions affect specified I_T nexus(es). The specified I_T nexus(es) are based on the SERVICE ACTION RESERVATION KEY field, SPEC_I_PT bit, and ALL_TG_PT bits as described in table xx.

Table xx. SPEC_I_PT and ALL_TG_PT bits for PREEMPT and PREEMPT AND ABORT

SPEC_I_PT bit	ALL_TG_PT bit	SERVICE ACTION RESERVATION KEY set to zero and All Registrants type?	Specified I_T nexus(es)
0	any (1)	no	All registered I_T nexus(es) with the reservation key specified by the SERVICE ACTION RESERVATION KEY field.
		yes	All registered I_T nexus(es).
1	0	no	All registered I_T nexus(es) with the reservation key specified by the SERVICE ACTION RESERVATION KEY field involving: a) the initiator port(s) specified in additional parameter data; and b) the target port used for the PERSISTENT RESERVE OUT command.
		yes	All registered I_T nexus(es) involving: a) the initiator port(s) specified in additional parameter data; and b) the target port used for the PERSISTENT RESERVE OUT command.
1	1	no	All registered I_T nexuses with the reservation key specified by the SERVICE ACTION RESERVATION KEY field involving: a) the initiator port(s) specified in additional parameter data; and b) any target port.
		yes	All registered I_T nexuses involving: a) the initiator port(s) specified in additional parameter data; and b) any target port.
(1) the ALL_TG_PT bit has no effect for these service actions unless SPEC_I_PT bit is set to one			

[Editor's note: Alternative: make ALL_TG_PT unused for PREEMPT, or invert the ALL_TG_PT meaning for PREEMPT so both 00 and 10 mean all target ports while 01 and 11 mean only target port of the I_T nexus used by the PR OUT]

5.6.2.7.4.2 Failed persistent reservation preempt

If the preempting I_T nexus' 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 failed initiator port or failed SCSI initiator device), 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.

5.6.2.7.4.3 Preempting persistent reservations and registration handling

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 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(s) to be preempted;

bb) the SPEC_I_PT bit and ALL_TG_PT bit optionally selecting a subset of I_T nexuses 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 there is no persistent reservation, the device server shall return GOOD status without modifying any registrations or persistent reservations.

If the SERVICE ACTION RESERVATION KEY field, SPEC_I_PT bit, and ALL_TG_PT bit identify~~ies a persistent reservation holder~~ one or more persistent reservation holders (see 5.6.2.6), the device server shall perform a preempt by doing the following as an uninterrupted series of actions:

a) Release the persistent reservation for the persistent reservation holder(s) ~~identified by the SERVICE ACTION RESERVATION KEY~~;

b) Remove the registrations for all I_T nexuses identified by the SERVICE ACTION RESERVATION KEY~~,~~ field, SPEC_I_PT bit, and ALL_TG_PT bit, except the I_T nexus that ~~issued~~ processed 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 the ~~initiator port~~that of the I_T nexus that ~~issued~~ processed 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 ~~an~~ 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 and the task aborted status is not enabled (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.

5.6.2.7.4.4 Removing registrations

~~When a registered reservation key does not identify a persistent reservation holder (see 5.6.2.6), an~~ 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 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 ~~I~~ nexus; and

b) SERVICE ACTION RESERVATION KEY field set to match the reservation key of the registration(s) ~~being~~ to be removed.

If there is no persistent reservation, the device server shall return GOOD status without modifying any registrations or persistent reservations.

If there is a persistent reservation and if the SERVICE ACTION RESERVATION KEY field 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 I_T nexus or~~all I_T nexuses ~~identified~~ specified by the SERVICE ACTION RESERVATION KEY field, SPEC_I_PT bit, and ALL_TG_PT bit;

b) Ignore the contents of the SCOPE and TYPE fields;

c) Process tasks as defined in 5.6.1; and

d) Establish a unit attention condition for ~~any~~ every initiator port associated with ~~an~~ every I_T nexus that lost its registration other than the initiator port that sent the PERSISTENT RESERVE OUT command. The sense key shall be set to UNIT ATTENTION and the additional sense code shall be set to REGISTRATIONS PREEMPTED.

If a PERSISTENT RESERVE OUT with ~~a~~ PREEMPT ~~service action~~ or ~~a~~ PREEMPT AND ABORT service action ~~sets the SERVICE ACTION RESERVATION KEY field to a value that does not match any registered reservation key~~ specifies no I_T nexuses, the device server shall return a RESERVATION CONFLICT status.

It is not an error for a PERSISTENT RESERVE OUT with ~~a~~ PREEMPT ~~service action~~ or ~~a~~ PREEMPT AND ABORT service action to set the RESERVATION KEY and the SERVICE ACTION RESERVATION KEY to the same value and specify only the I_T nexus used for the PERSISTENT RESERVE OUT command~~, h~~. However, no unit attention condition is established for the initiator port that sent the PERSISTENT RESERVE OUT command.

5.6.2.7.5 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 5.6.2.7.4) except for the following additions. If no reservation conflict occurred, the device server shall perform the following uninterrupted series of actions:

a) If the TST field is 000b (see 7.4.6) and an ACA condition exists for initiator ports other than the initiator port associated with the I_T nexus that is 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 the ACA condition for initiator ports other than the initiator port associated with the I_T nexuses that are 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 5.6.2.7.4);

c) All tasks from the initiator port(s) associated with the preempted I_T nexuses (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 preempted I_T nexus (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) 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; and

e) For 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 associated with the I_T nexus ~~or I_T nexus~~(es) being preempted (see 6.13).

The actions described in the preceding list shall be performed for all specified 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 and the SERVICE ACTION RESERVATION KEY value is set to zero the device server shall abort all tasks for all registered I_T nexuses.~~

5.6.2.7.6 Clearing

Any application client may release a persistent reservation and remove all registrations from a device server by issuing a PERSISTENT RESERVE OUT command with CLEAR service action through a registered I_T nexus with the following parameter:

a) RESERVATION KEY field set to the value of the reservation key that is registered for the I_T ~~_L~~ nexus.

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;

b) Remove all registration(s) (see 5.6.2.4);

c) Ignore the contents of the SCOPE and TYPE fields;

d) Continue normal execution of any tasks from any I_T nexus that have been accepted by the device server as allowed (i.e., nonconflicting); and

e) Establish a unit attention condition for every initiator port associated with every registered I_T nexus~~es~~ 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 code 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.

5.6.2.8 Third party persistent reservations

Persistent reservations may be used to give temporary access to another initiator port (e.g., a copy manager supporting the EXTENDED COPY command).

The recommended steps are:

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 will use and move the persistent reservation to that I_T nexus;
5. Backup application sends the EXTENDED COPY command to the copy manager;
6. Copy manager accesses the logical unit (e.g., writes to the tape); and
7. When the EXTENDED COPY command completes, backup application uses the PREEMPT service action, specifying the reservation key and I_T nexus used by the copy manager, to regain ownership of the persistent reservation.

If the same reservation key is used for both the backup application and the copy manager, a preempt specifying that reservation key and not specifying an initiator port affects both the backup application's registration and the copy manager's registration, so the preempting application need not be aware of whether a copy manager is being used.

[6.11 PERSISTENT RESERVE IN command]
6.12 PERSISTENT RESERVE OUT command

6.12.1 PERSISTENT RESERVE OUT command introduction

The PERSISTENT RESERVE OUT command (see table 105) 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 105 — PERSISTENT RESERVE OUT command]

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 implemented, 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 ~~S~~service ~~A~~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 106.

Table 106 — PERSISTENT RESERVE OUT service action codes

Code	Name	Description	PRGENERATION FIELD INCREMENTED
00h	REGISTER	Register a reservation key with the device server (see 5.6.2.4) or unregister a reservation key (see 5.6.2.7.3).	yes
01h	RESERVE	Creates a persistent reservation having a specified SCOPE and TYPE (see 5.6.2.5). The SCOPE and TYPE of a persistent reservation are defined in 6.11.4.2 and 6.11.4.3.	no
02h	RELEASE	Releases the selected persistent reservation (see 5.6.2.7.2).	no
03h	CLEAR	Clears all reservation keys (i.e., registrations) and all persistent reservations (see 5.6.2.7.6).	yes
04h	PREEMPT	Preempts persistent reservations and/or removes registrations (see 5.6.2.7.4).	yes
05h	PREEMPT AND ABORT	Preempts persistent reservations and/or removes registrations and aborts all tasks for all preempted I_T nexuses (see 5.6.2.7.4 and 5.6.2.7.5).	yes
06h	REGISTER AND IGNORE EXISTING KEY	Register a reservation key with the device server (see 5.6.2.4) or unregister a reservation key (see 5.6.2.7.3).	yes
07h	REGISTER AND MOVE	Register a reservation key with the device server and move a persistent reservation (see 5.6.2.x).	yes
0~~7~~8h - 1Fh	Reserved		

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

6.12.3 PERSISTENT RESERVE OUT parameter list

The parameter list required to perform the PERSISTENT RESERVE OUT command is defined in table 107. All fields shall be sent on all PERSISTENT RESERVE OUT commands, even if the field is not required for the specified service action and scope values.

Table 107 — PERSISTENT RESERVE OUT parameter list

Byte\Bit	7	6	5	4	3	2	1	0
0	(MSB)	RESERVATION KEY						
7								(LSB)
8	(MSB)	SERVICE ACTION RESERVATION KEY						
15								(LSB)
16	(MSB)	SCOPE-SPECIFIC ADDRESS						
19								(LSB)
20	Rsvd	Rsvd	Rsvd	Rsvd	SPEC_I_PT	ALL_TG_PT	Rsvd	APTPL
21	Reserved							
22		Obsolete						
23								
24		Additional parameter data						
n								

The obsolete field in Bytes 22 and 23 was defined in a previous standard for use with an obsolete scope (see table 101). If the obsolete scope is not supported Bytes 22 and 23 should be zero.

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, except for:

a) The REGISTER AND IGNORE EXISTING KEY service action where the RESERVATION KEY field shall be ignored; and

b) The REGISTER service action for an unregistered I_T nexus where the RESERVATION KEY field shall contain zero.

Except as noted above, 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. Except as noted above, 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 information needed for ~~four service actions:~~ the REGISTER, REGISTER AND IGNORE EXISTING KEY, REGISTER AND MOVE, PREEMPT, and PREEMPT AND ABORT service actions. The SERVICE ACTION RESERVATION KEY field is ignored for all other service actions.

For the REGISTER ~~and~~, REGISTER AND IGNORE EXISTING KEY, and REGISTER AND MOVE service action~~s~~, the SERVICE ACTION RESERVATION KEY field contains:

a) The new reservation key to be registered ~~in place of the registered reservation key specified in the RESERVATION KEY field~~; or

b) Zero to unregister ~~the registered reservation key specified in the RESERVATION KEY field~~.

For the PREEMPT and PREEMPT AND ABORT service actions, the SERVICE ACTION RESERVATION KEY field contains the reservation key of:

a) The registrations to be removed; and

b) If the SERVICE ACTION RESERVATION KEY field ~~identifies~~ and any specify initiator ports parameter data identify a persistent reservation holder (see 5.6.2.6), persistent reservations that are to be preempted.

If the scope is an ELEMENT_SCOPE persistent reservation, the SCOPE-SPECIFIC ADDRESS field shall contain the element address, zero filled in the most significant bits to fit the field. If the service action is REGISTER, REGISTER AND IGNORE EXISTING KEY, REGISTER AND MOVE, or CLEAR, or if the scope is a LU_SCOPE persistent reservation, the SCOPE-SPECIFIC ADDRESS field shall be set to zero.

If the SPEC_I_PT (Specify Initiator Ports) bit is set to zero, the device server shall ignore the additional parameter data.

If the SPEC_I_PT bit is set to one, the additional parameter data shall include a list of transport IDs (see table 108).

For the REGISTER, REGISTER AND IGNORE EXISTING KEY, REGISTER AND MOVE service actions, the SPEC_I_PT bit usage is described in 5.6.2.4. For the PREEMPT and PREEMPT AND ABORT service actions, the SPEC_I_PT bit usage is described in 5.6.2.7.4.1.

~~and shall apply the registration only to the I_T nexus that sent the PERSISTENT RESERVE OUT command. If the SPEC_I_PT bit is set to one for the REGISTER or REGISTER AND IGNORE EXISTING KEY service actions, the additional parameter data shall include a list of transport IDs (see table 108) and the device server shall apply the registration to the I_T nexus for each initiator port specified by a TransportID. If a registration fails for any initiator port (e.g., if the SCSI target device does not have enough resources available to hold the registration information), none of the other registrations shall be made.~~

[Table 108 — PERSISTENT RESERVE OUT specify initiator ports additional parameter data]

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

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 additional parameter list bytes specified by the TRANSPORTID PARAMETER DATA LENGTH field; or

b) If the value in the TRANSPORTID PARAMETER DATA LENGTH field results in the truncation of a TransportID.

The format of a TransportID is specified in 7.5.4.

The ALL_TG_PT (All Target Ports) bit is valid only for the REGISTER, ~~and~~ REGISTER AND IGNORE EXISTING KEY, REGISTER AND MOVE, PREEMPT, and PREEMPT AND ABORT service actions, and shall be ignored for all other service actions. Support for the ALL_TG_PT bit is optional. For the REGISTER, REGISTER AND IGNORE EXISTING KEY, REGISTER AND MOVE service actions, the ALL_TG_PT bit usage is described in 5.6.2.4. For the PREEMPT and PREEMPT AND ABORT service actions, the ALL_TG_PT bit usage is described in 5.6.2.7.4.1.

~~If the device server receives a REGISTER or REGISTER AND IGNORE EXISTING KEY service action with the ALL_TG_PT bit set to one, it shall create the specified registration on all target ports in the target device (i.e., as if the same registration request had been received individually through each target port). If the device server receives a REGISTER or REGISTER AND IGNORE EXISTING KEY service action with the ALL_TG_PT bit set to zero, it shall apply the registration only to the target port through which the PERSISTENT RESERVE OUT command was received.~~

The APTPL (Activate Persist Through Power Loss) bit is valid only for the REGISTER, ~~and~~ REGISTER AND IGNORE EXISTING KEY, and REGISTER AND MOVE service actions, and shall be ignored for all other service actions. Support for an APTPL bit equal to one is optional. If a device server that does not support an APTPL bit set to one receives that value ~~in a REGISTER or a REGISTER AND IGNORE EXISTING KEY service action~~, 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 PARAMETER LIST. Usage of the APTPL bit is described in 5.6.2.2.

~~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 5.6.2.4).~~

~~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 5.6.2.2).~~

[Editor's note: the preceding two paragraphs are more precisely covered in the model section]

Table 109 summarizes which fields are set by the application client and interpreted by the device server for each service action and scope value.

Table 109 — PERSISTENT RESERVE OUT service actions and valid parameters (part 1 of 2)

Service action	Allowed SCOPE	TYPE	RESERVATION KEY	SERVICE ACTION RESERVATION KEY	SCOPE-SPECIFIC ADDRESS
REGISTER	ignored	ignored	valid	valid	ignored
REGISTER AND IGNORE EXISTING KEY	ignored	ignored	ignored	valid	ignored
REGISTER AND MOVE	ignored	ignored	valid	valid	ignored
RESERVE	LU_SCOPE ELEMENT_SCOPE	valid	valid	ignored	ignored valid (element)
RELEASE	ignored	valid	valid	ignored	ignored valid (element)
CLEAR	LU_SCOPE ELEMENT_SCOPE	ignored	valid	ignored	ignored
PREEMPT	LU_SCOPE ELEMENT_SCOPE	valid	valid	valid	ignored valid (element)
PREEMPT AND ABORT	LU_SCOPE ELEMENT_SCOPE	valid	valid	valid	ignored valid (element)

Table 109 — PERSISTENT RESERVE OUT service actions and valid parameters (part 2 of 2)

Service action	Allowed SCOPE	APTPL	ALL_TG_PT	SPEC_I_PT
REGISTER	ignored	valid	valid	valid
REGISTER AND IGNORE EXISTING KEY	ignored	valid	valid	valid
REGISTER AND MOVE	ignored	valid	valid	valid
RESERVE	all	ignored	ignored	ignored
RELEASE	all	ignored	ignored	ignored
CLEAR	ignored	ignored	ignored	ignored
PREEMPT	all	ignored	~~ignored~~valid	~~ignored~~valid
PREEMPT AND ABORT	all	ignored	~~ignored~~valid	~~ignored~~valid