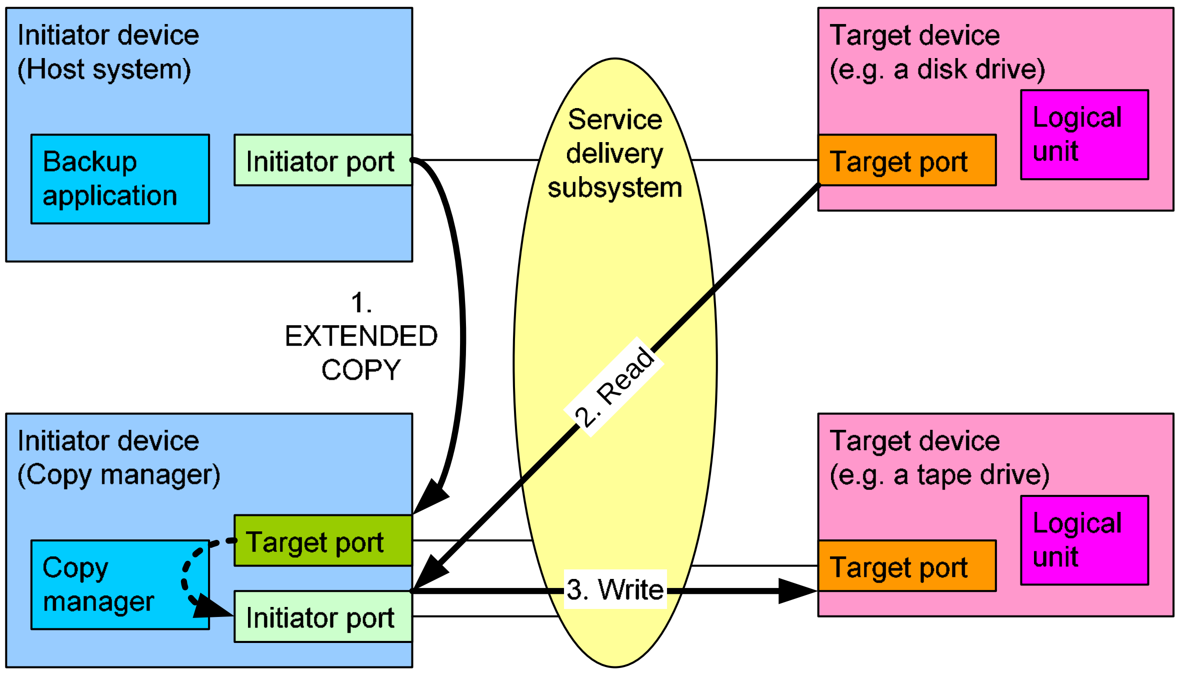
To:	T10 Technical Committee
From:	Rob Elliott, HP (elliott@hp.com)
Date:	3 October 2003
Subject:	T10/03-337r0 SPC-3 Third party persistent reservations

Revision History
Revision 0 (3 October 2003) first revision

Related Documents
03-231r0 Two Persistent Reservations problems - latest information (Roger Cummings, Veritas)
03-233r0 New PR Out Service Action Proposal (Roger Cummings, Veritas)
03-321r0 Persistent Reservations Proposals (Roger Cummings, Veritas)
spc3r14 – SCSI Primary Commands – 3 revision 14

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.


Initiator device (Host system)
Backup application
Initiator port
Service delivery subsystem
Target device (e.g. a disk drive)
Target port
Logical unit
1. EXTENDED COPY
2. Read
Initiator device (Copy manager)
Target port
Copy manager
Initiator port
3. Write
Target device (e.g. a tape drive)
Target port
Logical unit

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

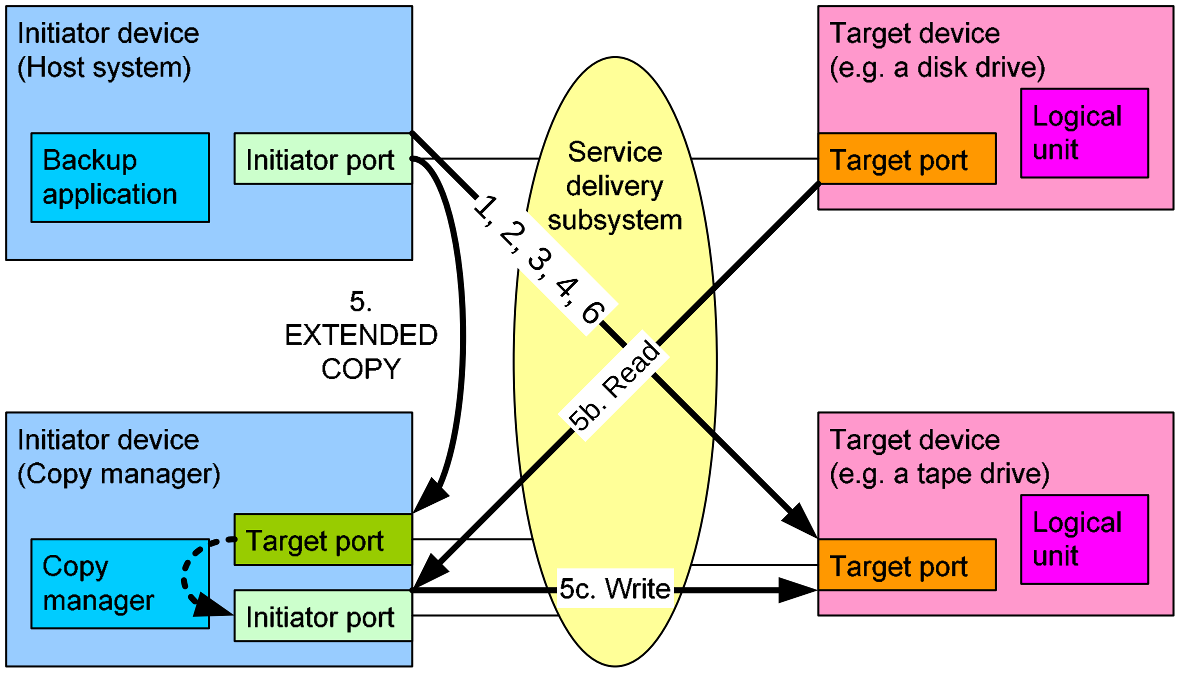
Other proposals
03-233 proposed a MOVE service action, specifying the key to move. However, isn't precise enough if more than one I_T is registered with the same key. The Specify Initiator ports feature can be used to fix this.

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 key.

Proposal

Add GIVE and TAKE bits to the REGISTER service actions to move the reservation.

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
5. Send EXTENDED COPY to copy manager; wait for completion
6. Unregister copy manager and take back reservation with PR OUT/REGISTER



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. Give and take

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

- SPEC_I_PT bit set to 1
- parameter data specifying the copy manager's initiator port (exactly one TransportID)
- (new) GIVE bit set to 1
- SERVICE ACTION RESERVATION KEY set to a key for the copy manager
- RESERVATION KEY set to its own key (normal rules for these service actions)

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

It is an error to use the GRANT bit 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 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 key as the backup application, so if the 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 REGISTER or REGISTER AND IGNORE EXISTING KEY service action with:

- SPEC_I_PT bit set to 1
- parameter data specifying the copy manager's initiator port
- (new) TAKE bit set to 1
- SERVICE ACTION RESERVATION KEY set to 00000000h
- 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.

It is an error to use the TAKE bit if the specified initiator is not a reservation holder or if more than one initiator port is specified in the parameter list.

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

<u>Suggested Changes</u>

5.6 Reservations

5.6.1 Reservations overview

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

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

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

Reservation restrictions are placed on commands as a result of access qualifiers associated with the type of reservation.

The details of which commands are allowed under what types of reservations are described in table 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~~<u>that are not persistent reservation holders (see 5.6.2.6)</u> 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~~ <u>that are not persistent reservation holders</u> or from I_T nexuses not registered when a registrants only or all registrants type persistent reservation is present shall not be performed and the device server shall terminate the command with a RESERVATION CONFLICT status.

Commands from I_T nexuses ~~holding a reservation~~ 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]

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 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, holds the persistent reservation. The reservation key is used in the PERSISTENT RESERVE OUT command to register an I_T nexus, to verify the I_T nexus issuing 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 key shall be maintained for each I_T nexus, regardless of the 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 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 para seems to hint that a single PR OUT might affect more than one LU.]

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 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 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 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 is a good time to address this.]

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 SPEC_I_PT bit (see 6.12.1) controls which I_T nexus(es) are registered.

If the I_T nexus used to send the PERSISTENT RESERVE OUT command~~has not yet established a reservation key or the reservation key and registration have been removed~~is not currently registered, the registration ~~is~~ may be 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 used to send the PERSISTENT RESERVE OUT command has an established registration ~~it~~ the application client ~~may change its reservation key. This is~~, the registration may be 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 for the I_T nexus used to send the PERSISTENT RESERVE OUT command 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 or 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 or REGISTER AND IGNORE EXISTING KEY service action is sent when there is no established registration, a new registration shall be established.

If a PERSISTENT RESERVE OUT command with REGISTER or a REGISTER AND IGNORE EXISTING KEY 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 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 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 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 register with the same reservation key. An application client may use the same reservation key for other I_T nexuses and logical units.

5.6.2.5 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.

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]

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

5.6.2.8 Third party persistent reservations

The GIVE and TAKE bits in the PERSISTENT RESERVE OUT command parameter data for the REGISTER and REGISTER AND IGNORE EXISTING KEY service actions may be used to temporarily move a reservation from one I_T nexus (e.g., serving a backup application) to another I_T nexus (e.g., a copy manager supporting the EXTENDED COPY command).

The recommended steps are:
1. Backup application registers an I_T nexus to a logical unit (e.g., containing a tape drive);
2. Backup application establishes a persistent reservation with the exclusive access type;
3. Backup application prepare the logical unit for access (e.g., ensures the tape is loaded and rewound);
4. Backup application registers the I_T nexus of the copy manager with the same reservation key it is using and gives that I_T nexus its persistent reservation;
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 EXTENDED COPY completes, backup application takes back the persistent reservation.
[Editor's note: this list may be too useful to go into the standard**]**

If the same key is used, a preempt will affect both the backup application's registration and the copy manager's registration.

The GIVE and TAKE bits may be used with any reservation type.

[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 Service Actions

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

The PERSISTENT RESERVE OUT command service actions are defined in table 106.

[Table 106 — PERSISTENT RESERVE OUT service action codes]

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

	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	GIVE	TAKE	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, 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 service action, 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 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, or CLEAR or if the scope is a LU_SCOPE persistent reservation, the SCOPE-SPECIFIC ADDRESS field shall be set to zero.

The GIVE bit and TAKE bit are used to move reservations between I_T nexuses (i.e., perform third party persistent reservations).

If:

a) the GIVE bit is set to one;
b) the service action is REGISTER or REGISTER AND IGNORE EXISTING KEY;
c) the SPEC_I_PT bit is set to one;
d) the additional parameter data contains exactly one TransportID; and
e) the I_T nexus that sent the PERSISTENT RESERVE OUT command is a reservation holder;

then the device server shall:

1) register the I_T nexus specified by the additional parameter data (see 5.6.2.4);
2) remove the reservation from the I_T nexus that sent the PERSISTENT RESERVE OUT command to the specified I_T nexus; and
3) grant the reservation to the specified I_T nexus.

NOTE: The I_T nexus that sent the PERSISTENT RESERVE OUT command remains registered. if the type of registration is All Registrants, then the I_T nexus remains a reservation holder. If the type of reservation is Registrants Only, then the I_T nexus maintains access.

If any of the conditions are not true, the GIVE bit shall be set to zero.

If:

a) the TAKE bit is set to one;
b) the service action is REGISTER or REGISTER AND IGNORE EXISTING KEY;
c) the SERVICE ACTION RESERVATION KEY field is set to zero;
the SPEC_I_PT bit is set to one;
d) the additional parameter data contains exactly one TransportID;
e) the I_T nexus that sent the PERSISTENT RESERVE OUT command is a reservation holder;

then the device server shall:

1) remove the reservation from the I_T nexus specified by the additional parameter data;
2) grant the reservation to the I_T nexus that sent the PERSISTENT RESERVE OUT command to the specified I_T nexus; and
3) unregister the specified I_T nexus (see 5.6.2.7.3).

If any of the conditions are not true, the TAKE bit 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 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 service actions, and shall be ignored for all other service actions. Support for the ALL_TG_PT bit is optional. 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 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.

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).

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