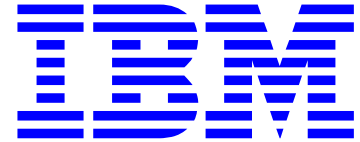


To: INCITS Technical Committee T10
From: Kevin Butt
Date: Printed Wednesday, January 23, 2008 10:01 am
Document: T10/08-025r3 — Persistent Reservations - Team



Revisions

1. 08-025r0 Initial revision (10 December 2007)
2. 08-025r1 Incorporated concepts agreed to on Group Reservations call 07 January 2008 including changing the name from Group to Team. No change bars.
3. 08-025r2 Incorporate feedback received offline. Shown with change bars.
4. 08-025r3 Incorporate feedback from January CAP meeting.

Introduction

Please see the presentation in [08-024r2](#) for the detailed introduction. There is a need for protecting a team of coordinating hosts against a third party joining their reservation. This proposal attempts to solve that by adding a team reservation.

Key:

[Added Text](#)

~~Deleted Text~~

Proposal

5.6 Reservations

5.6.1 Persistent Reservations overview

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

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

The persistent reservations mechanism allows multiple application clients communicating through multiple I_T nexuses to preserve reservation operations across SCSI initiator device failures, which usually involve logical unit resets and involve I_T nexus losses. Persistent reservations persist across recovery actions. 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 its 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 SCSI target device 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 accessing a logical unit.

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

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

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

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

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

See table 131 in 6.14.2 for a list of PERSISTENT RESERVE OUT service actions. See table 119 in 6.13.1 for a list of PERSISTENT RESERVE IN service actions.

The scope (see 6.13.3.3) of a persistent reservation shall be the entire logical unit.

The type (see 6.13.3.4) of a persistent reservation defines the selected set of I_T nexuses for which the persistent reservation places restrictions on commands.

The details of which commands are allowed under what types of reservations are described in table 1.

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

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

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

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

A command shall be checked for reservation conflicts before the task containing that command enters the enabled task state. Once a task has entered the enabled task state, the command that comprises the task shall not be terminated with a RESERVATION CONFLICT due to a subsequent reservation.

For each command, this standard or a command standard (see 3.1.17) defines the conditions that result in RESERVATION CONFLICT. Command standards define the conditions either in the device model or in the descriptions ~~each~~ of each specific command.

Table 1 — SPC commands that are allowed in the presence of various reservations (part 1 of 3)

| Command | Addressed logical unit has this type of persistent reservation held by another I_T nexus | | | | |
|--------------------|--|------------------------|---|---|------------------------|
| | From any I_T nexus <u>that is not a reservation holder</u> | | From registered permitted ^c I_T nexus (RR all-types) | From not registered permitted ^c I_T nexus | |
| | Write Excl_ <u>ET</u> | Excl Access <u>-ET</u> | | Write Excl_ <u>-RR</u> | Excl Access <u>-RR</u> |
| ACCESS CONTROL IN | Allowed | Allowed | Allowed | Allowed | Allowed |
| ACCESS CONTROL OUT | Allowed | Allowed | Allowed | Allowed | Allowed |
| CHANGE ALIASES | Conflict | Conflict | Allowed | Conflict | Conflict |
| EXTENDED COPY | Conflict | Conflict | Allowed | Conflict | Conflict |
| INQUIRY | Allowed | Allowed | Allowed | Allowed | Allowed |
| LOG SELECT | Conflict | Conflict | Allowed | Conflict | Conflict |
| LOG SENSE | Allowed | Allowed | Allowed | Allowed | Allowed |

Key: **Excl**=Exclusive, **RR**=Registrants Only or All Registrants, **ET**=Exclusive or Team, <> Not Equal

^a Exceptions to the behavior of the RESERVE and RELEASE commands described in SPC-2 are defined in 5.6.4.

^b Logical units claiming compliance with previous versions of this standard (e.g., SPC-2, SPC-3) may return RESERVATION CONFLICT in this case. Logical units may report whether certain commands are allowed in ALLOW COMMANDS field of the parameter data returned by the PERSISTENT RESERVE IN command with REPORT CAPABILITIES service action (see 6.13.4).

^c Permitted I_T nexus: If the type of reservation is an RR, then any registered I_T nexus. If the type of reservation is a Team type reservation, then a reservation holder.

Table 1 — SPC commands that are allowed in the presence of various reservations (part 2 of 3)

| Command | Addressed logical unit has this type of persistent reservation held by another I_T nexus | | | | |
|--|--|------------------------|---|---|------------------------|
| | From any I_T nexus <u>that is not a reservation holder</u> | | From registered permitted ^c I_T nexus (RR-all-types) | From not registered permitted ^c I_T nexus | |
| | Write Excl_ <u>ET</u> | Excl Access <u>-ET</u> | | Write Excl_ <u>-RR</u> | Excl Access <u>-RR</u> |
| MANAGEMENT PROTOCOL IN | Allowed | Conflict | Allowed | Allowed | Conflict |
| MANAGEMENT PROTOCOL OUT | Conflict | Conflict | Allowed | Conflict | Conflict |
| MODE SELECT(6) / MODE SELECT(10) | Conflict | Conflict | Allowed | Conflict | Conflict |
| MODE SENSE(6) / MODE SENSE(10) | Allowed ^b | Conflict | Allowed | Allowed ^b | Conflict |
| PERSISTENT RESERVE IN | Allowed | Allowed | Allowed | Allowed | Allowed |
| PERSISTENT RESERVE OUT | see table 2 | | | | |
| READ ATTRIBUTE | Allowed ^b | Conflict | Allowed | Allowed ^b | Conflict |
| READ BUFFER | Allowed ^b | Conflict | Allowed | Allowed ^b | Conflict |
| READ MEDIA SERIAL NUMBER | Allowed | Allowed | Allowed | Allowed | Allowed |
| RECEIVE COPY RESULTS | Conflict | Conflict | Allowed | Conflict | Conflict |
| RECEIVE DIAGNOSTIC RESULTS | Allowed ^b | Conflict | Allowed | Allowed ^b | Conflict |
| RELEASE(6)/ RELEASE(10) | As defined in SPC-2 ^a | | | | |
| REPORT ALIASES | Allowed | Allowed | Allowed | Allowed | Allowed |
| REPORT IDENTIFYING INFORMATION | Allowed | Allowed | Allowed | Allowed | Allowed |
| REPORT LUNS | Allowed | Allowed | Allowed | Allowed | Allowed |
| REPORT PRIORITY | Allowed | Allowed | Allowed | Allowed | Allowed |
| REPORT SUPPORTED OPERATION CODES | Allowed ^b | Conflict | Allowed | Allowed ^b | Conflict |
| REPORT SUPPORTED TASK MANAGEMENT FUNCTIONS | Allowed ^b | Conflict | Allowed | Allowed ^b | Conflict |
| REPORT TARGET PORT GROUPS | Allowed | Allowed | Allowed | Allowed | Allowed |
| REPORT TIMESTAMP | Allowed | Allowed | Allowed | Allowed | Allowed |
| Key: Excl =Exclusive, RR =Registrants Only or All Registrants, ET=Exclusive or Team , <> Not Equal | | | | | |
| <p>^a Exceptions to the behavior of the RESERVE and RELEASE commands described in SPC-2 are defined in 5.6.4.</p> <p>^b Logical units claiming compliance with previous versions of this standard (e.g., SPC-2, SPC-3) may return RESERVATION CONFLICT in this case. Logical units may report whether certain commands are allowed in ALLOW COMMANDS field of the parameter data returned by the PERSISTENT RESERVE IN command with REPORT CAPABILITIES service action (see 6.13.4).</p> <p>^c <u>Permitted I_T nexus: If the type of reservation is an RR, then any registered I_T nexus. If the type of reservation is a Team type reservation, then a reservation holder.</u></p> | | | | | |

Table 1 — SPC commands that are allowed in the presence of various reservations (part 3 of 3)

| Command | Addressed logical unit has this type of persistent reservation held by another I_T nexus | | | | |
|---|--|------------------------|---|---|------------------------|
| | From any I_T nexus <u>that is not a reservation holder</u> | | From registered permitted ^c I_T nexus (RR all types) | From not registered permitted ^c I_T nexus | |
| | Write Excl_ <u>ET</u> | Excl Access <u>-ET</u> | | Write Excl_ <u>-RR</u> | Excl Access <u>-RR</u> |
| REQUEST SENSE | Allowed | Allowed | Allowed | Allowed | Allowed |
| RESERVE(6) / RESERVE(10) | As defined in SPC-2 ^a | | | | |
| SECURITY PROTOCOL IN | Allowed | Conflict | Allowed | Allowed | Conflict |
| SECURITY PROTOCOL OUT | Conflict | Conflict | Allowed | Conflict | Conflict |
| SEND DIAGNOSTIC | Conflict | Conflict | Allowed | Conflict | Conflict |
| SET IDENTIFYING INFORMATION | Conflict | Conflict | Allowed | Conflict | Conflict |
| SET PRIORITY | Conflict | Conflict | Allowed | Conflict | Conflict |
| SET TARGET PORT GROUPS | Conflict | Conflict | Allowed | Conflict | Conflict |
| SET TIMESTAMP | Conflict | Conflict | Allowed | Conflict | Conflict |
| TEST UNIT READY | Allowed ^b | Allowed ^b | Allowed | Allowed ^b | Allowed ^b |
| WRITE ATTRIBUTE | Conflict | Conflict | Allowed | Conflict | Conflict |
| WRITE BUFFER | Conflict | Conflict | Allowed | Conflict | Conflict |
| Key: Excl =Exclusive, RR =Registrants Only or All Registrants, ET=Exclusive or Team. <> Not Equal | | | | | |
| ^a Exceptions to the behavior of the RESERVE and RELEASE commands described in SPC-2 are defined in 5.6.4. ^b Logical units claiming compliance with previous versions of this standard (e.g., SPC-2, SPC-3) may return RESERVATION CONFLICT in this case. Logical units may report whether certain commands are allowed in ALLOW COMMANDS field of the parameter data returned by the PERSISTENT RESERVE IN command with REPORT CAPABILITIES service action (see 6.13.4). ^c <u>Permitted I_T nexus: If the type of reservation is an RR, then any registered I_T nexus. If the type of reservation is a Team type reservation, then a reservation holder.</u> | | | | | |

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

| Command Service Action | Addressed logical unit 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 | Conflict | Conflict |
| RELEASE | Allowed ^a | Conflict |
| RESERVE | Conflict | Conflict |

^a The reservation is not released (see 5.6.11.2).

The time at which a reservation is established with respect to other tasks being managed by the device server is vendor specific. Successful completion of a reservation command indicates that the new reservation is established. A reservation may apply to some or all of the tasks in the task set before the completion of the reservation command. The reservation shall apply to all tasks received by the device server after successful completion of the reservation command. Any persistent reserve service action shall be performed as a single indivisible event.

Multiple persistent reserve service actions may be present in the task set at the same time. The order of processing of such service actions is defined by the task set management requirements defined in SAM-4, but each is processed as a single indivisible command without any interleaving of actions that may be required by other reservation commands.

5.6.2 Team reservations

5.6.2.1 Team reservation overview

There may be a need for a team of application clients to share access to a device server amongst the team while restricting access to other application clients. This is accomplished by using a team reservation. When a team reservation is in effect, all application clients that are not members of the team will have commands rejected with a reservation conflict (see 5.6.1) while application clients that are members of the team will have commands processed without a reservation conflict.

5.6.2.2 Creating a team reservation

A team reservation is created when the device server receives a PERSISTENT RESERVE OUT command using the RESERVE service action (see 5.6.9) with the TYPE field set to Write Exclusive – Team or Exclusive Access – Team and using the team reservation descriptor (see xxx). The TEAM RESERVATION KEY field (see xxx) contains a value that is saved by the device server as the team reservation key which is associated with the team reservation created by this command. The application client should guarantee that the team reservation key is unique and unlikely to be used by others. When creating a team reservation, the application client may send an optional list of team members using TransportIDs (see 6.14.1). When a list of team members is received with a PERSISTENT

[RESERVE OUT command that creates a team reservation, the device server joins each I_T nexus indicated by the team member list to the team reservation by performing the actions described in clause 5.6.2.3 for each I_T nexus.](#)

[5.6.2.3 Joining a team reservation](#)

[A team reservation may be joined by any application client that uses a team reservation key that matches the team reservation key used when the team reservation was established. How an application client determines the team reservation key is outside the scope of this standard. An application client that is joined to a team reservation is a member of the team reservation.](#)

5.6.3 Third party persistent reservations

Except for all registrants type reservations [and team type reservations](#), a reservation holder (see 5.6.10) may move the persistent reservation to a third party (e.g., a copy manager supporting the EXTENDED COPY command) using the REGISTER AND MOVE service action (see 5.6.8). A copy manager supporting the EXTENDED COPY command may be instructed to move the persistent reservation to a specified I_T nexus using the third party persistent reservations source I_T nexus segment descriptor (see 6.3.7.19).

5.6.4 Exceptions to SPC-2 RESERVE and RELEASE behavior

[Editor's Note 1: No changes to this section.](#)

5.6.5 Preserving persistent reservations and registrations

5.6.5.1 Preserving persistent reservations and registrations through power loss

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

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

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

- a) For SCSI transport protocols where initiator port names (see 3.1.53) are required, the initiator port name; otherwise, the initiator port identifier (see 3.1.52);
- b) Reservation key; and
- c) Indication of the target port to which the registration was applied.

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

- a) For SCSI transport protocols where initiator port names are required, the initiator port name; otherwise, the initiator port identifier;
- b) Reservation key;
- c) [Team reservation key](#);
- d) [If the reservation is a team reservation, then the reservation team members](#);
- e) Scope;
- f) Type; and
- g) Indication of the target port through which the reservation was established.

NOTE 1 - The scope of a persistent reservation is always LU_SCOPE (see 6.13.3.3). For an all registrants type persistent reservation, only the scope and type need to be preserved.

5.6.5.2 Nonvolatile memory considerations for preserving persistent reservations and registrations

[Editor's Note 2: No change to this section](#)

5.6.6 Finding persistent reservations and reservation keys

[Editor's Note 3: The model section has nothing on reporting capabilities and I have not proposed adding it.](#)

5.6.6.1 Summary of commands for finding persistent reservations and reservation keys

[Editor's Note 4: No changes to this section.](#)

5.6.6.2 Reporting reservation keys

[Editor's Note 5: No change to this section.](#)

5.6.6.3 Reporting the persistent reservation

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

In response to a PERSISTENT RESERVE IN command with READ RESERVATION service action the device server shall report the following information for the persistent reservation, if any:

- a) The current PRgeneration value (see 6.13.2);
- b) The registered reservation key, if any, associated with the I_T nexus that holds the persistent reservation (see 5.6.10). If the persistent reservation is an all registrants type [or a team type](#), the registered reservation key reported shall be zero; and
- c) The scope and type of the persistent reservation, if any.

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

5.6.6.4 Reporting full status

Editor's Note 6: No change. There will be one descriptor for each I_T nexus and each I_T nexus that is part of the team will set the R HOLDER bit to one.

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

In response to a PERSISTENT RESERVE IN command with READ FULL STATUS service action the device server shall report the current PRgeneration value (see 6.13.2) and, for every I_T nexus that is currently registered, the following information:

- a) The registered reservation key;
- b) Whether the I_T nexus is a persistent reservation holder;
- c) If the I_T nexus is a persistent reservation holder, the scope and type of the persistent reservation;
- d) The relative target port identifier identifying the target port of the I_T nexus; and
- e) A TransportID identifying the initiator port of the I_T nexus.

5.6.7 Registering

Editor's Note 7: No change to this section.

5.6.8 Registering and moving the reservation

The PERSISTENT RESERVE OUT command REGISTER AND MOVE service action is used to register a specified I_T nexus (see table 3) and move the reservation to that I_T nexus.

Table 3 — Register behaviors for a REGISTER AND MOVE service action

| Command I_T nexus status | Parameter list fields ^a | | | Results |
|---|--|--------------------------------|--------|---|
| | RESERVATION KEY | SERVICE ACTION RESERVATION KEY | UNREG | |
| received on an unregistered I_T nexus | ignore | ignore | ignore | Return RESERVATION CONFLICT status. ^d |
| received on the registered I_T nexus of reservation holder | Not equal to I_T nexus reservation key | ignore | ignore | Return RESERVATION CONFLICT status. |
| | Equal to I_T nexus reservation key | zero | ignore | Return CHECK CONDITION status. ^b |
| | | non-zero ^c | zero | The I_T nexus on which PERSISTENT RESERVE OUT command was received shall remain registered. See this subclause for the registration and the move specifications. |
| | | | one | The I_T nexus on which PERSISTENT RESERVE OUT command was received shall be unregistered (see 5.6.11.3) upon completion of command processing. See this subclause for the registration and the move specifications. |
| received on a registered I_T nexus that is not the reservation holder | ignore | ignore | ignore | Return RESERVATION CONFLICT status. |

^a For requirements regarding other parameter list fields not shown in this table see 6.14.4.
^b The sense key shall be set to ILLEGAL REQUEST, and the additional sense code shall be set to INVALID FIELD IN PARAMETER LIST. Devices compliant with previous versions of this standard may return an additional sense code set to INVALID FIELD IN CDB.
^c The application client and backup application should use the same reservation key.
^d Devices compliant with previous versions of this standard may return CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action is attempted, but there are insufficient device server resources to complete the operation, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INSUFFICIENT REGISTRATION RESOURCES.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action is received and the established persistent reservation is a Write Exclusive - All Registrants type ~~or~~ an Exclusive Access - All Regis-

trants type, [a Write Exclusive - Team type, or an Exclusive Access - Team type](#) reservation, then the command shall be terminated with RESERVATION CONFLICT status.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action is received and there is no persistent reservation established, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

If a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action specifies a TransportID that is the same as the initiator port of the I_T nexus on which the command received, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

In response to a PERSISTENT RESERVE OUT command with a REGISTER AND MOVE service action the device server shall perform a register and move by doing the following as an uninterrupted series of actions:

- a) Process the APTPL bit;
- b) Ignore the contents of the SCOPE and TYPE fields;
- c) Associate the reservation key specified in the SERVICE ACTION RESERVATION KEY field with the I_T nexus specified as the destination of the register and move, where:
 - A) The I_T nexus is specified by the TransportID and the RELATIVE TARGET PORT IDENTIFIER field (see 6.14.4); and
 - B) Regardless of the TransportID format used, the association for the initiator port is based on either the initiator port name (see 3.1.53) on SCSI transport protocols where port names are required or the initiator port identifier (see 3.1.52) on SCSI transport protocols where port names are not required;
- d) Register the reservation key specified in the SERVICE ACTION RESERVATION KEY field;
- e) Retain the reservation key specified in the SERVICE ACTION RESERVATION KEY field and associated information;
- f) Release the persistent reservation for the persistent reservation holder (i.e., the I_T nexus on which the command was received);
- g) Move the persistent reservation to the specified I_T nexus using the same scope and type as the persistent reservation released in item f); and
- h) If the UNREG bit is set to one, unregister (see 5.6.11.3) the I_T nexus on which PERSISTENT RESERVE OUT command was received.

It is not an error for a REGISTER AND MOVE service action to register an I_T nexus that is already registered with the same reservation key or a different reservation key.

5.6.9 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 with the logical unit for the I_T nexus; ~~and~~
- b) TYPE field and SCOPE field set to the persistent reservation being created; ~~and~~
- c) [if the type is one of the team types of reservation, then the TEAM RESERVATION KEY field set to the value being used by the team to specify membership in the team.](#)

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

[If a team reservation is in effect and if the device server receives a PERSISTENT RESERVE OUT command with RESERVE service action where the TYPE field, the SCOPE field, and the TEAM RESERVATION KEY field contain the same values as the existing type, scope, and team reservation key from an I_T nexus that is not a persistent reser-](#)

vation holder, then if there are enough resources, the device server shall join that I_T nexus to the team reservation and shall return a GOOD status.

If the device server receives a PERSISTENT RESERVE OUT command from an I_T nexus other than a persistent reservation holder (see 5.6.10) 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, or TEAM RESERVATION KEY if any of an existing persistent reservation, 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 SPEC_I_PT bit is set to zero and the TYPE field, ~~and~~ the SCOPE field, and the TEAM RESERVATION KEY field if any contain the same values as the existing type, ~~and~~ scope, and team reservation key if any 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.

5.6.10 Persistent reservation holder

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

- a) For a persistent reservation of the type Write Exclusive – All Registrants or Exclusive Access – All Registrants, the persistent reservation holder is any registered I_T nexus; ~~or~~
- b) For a persistent reservation of the type Write Exclusive – Team or Exclusive Access – Team, the persistent reservation holder is any I_T nexus that is a member of the team (see x.x.x):
- or
- c) For all other persistent reservation types, the persistent reservation holder is the I_T nexus:
 - A) For which the reservation was established with a PERSISTENT RESERVE OUT command with REGISTER service action, REGISTER AND IGNORE EXISTING KEY service action, PREEMPT service action, or PREEMPT AND ABORT service action; or
 - B) To which the reservation was moved by a PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action.

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

- ~~a) For a persistent reservation of the type Write Exclusive – All Registrants or Exclusive Access – All Registrants, the reservation key shall be set to zero; or~~
- a) The reservation key shall be set to zero for a persistent reservation of the type:
 - A) Write Exclusive – All Registrants;
 - B) Exclusive Access – All Registrants;
 - C) Write Exclusive – Team; or
 - D) Exclusive Access – Team;
- 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, and TEAM RESERVATION KEY if any fields that match those of the persistent reservation (see 5.6.9).

A persistent reservation holder is allowed to release the persistent reservation using the PERSISTENT RESERVE OUT command with RELEASE service action (see 5.6.11.2).

If the registration of the persistent reservation holder is removed (see 5.6.11.1), the reservation shall be released. If the persistent reservation holder is more than one I_T nexus, the reservation shall not be released until the registrations for all persistent reservation holder I_T nexuses are removed.

5.6.11 Releasing persistent reservations and removing registrations

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

An application client may release or preempt the persistent reservation 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 with the logical unit for that I_T nexus:

- a) A PERSISTENT RESERVE OUT command with RELEASE service action from a persistent reservation holder (see 5.6.11.2);
- b) A PERSISTENT RESERVE OUT command with PREEMPT service action specifying the reservation key of the persistent reservation holder or holders (see 5.6.11.4);
- c) A PERSISTENT RESERVE OUT command with PREEMPT AND ABORT service action specifying the reservation key of the persistent reservation holder or holders (see 5.6.11.5);
- d) A PERSISTENT RESERVE OUT command with CLEAR service action (see 5.6.11.6); or
- e) If the I_T nexus is the persistent reservation holder and the persistent reservation is not an all registrants type [or a team type](#), then 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 5.6.11.3).

Table 4 defines processing for a persistent reservation released or preempted by an application client based on the reservation type.

Table 4 — Processing for a released or preempted persistent reservation

| Reservation Type | Processing |
|---|--|
| Write Exclusive – Registrants Only or Exclusive Access – Registrants Only | When the persistent reservation holder (see 5.6.10) of this reservation type becomes unregistered the persistent reservation shall be released. |
| Write Exclusive – All Registrants or Exclusive Access – All Registrants | This persistent reservation shall be released when the registration for the last registered I_T nexus is removed or when the type or scope is changed. |
| Write Exclusive – Team or Exclusive Access – Team | This persistent reservation shall be released when the registration for the last reservation holder (see clause 5.6.10) is removed or when the type or scope is changed. |
| Write Exclusive or Exclusive Access | When the persistent reservation holder of this reservation type becomes unregistered the persistent reservation shall be released. |

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 with the logical unit for that I_T nexus:

- a) A PERSISTENT RESERVE OUT command with PREEMPT service action with the SERVICE ACTION RESERVATION KEY field set to the reservation key (see 5.6.11.4) 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 5.6.11.5) to be removed;
- c) A PERSISTENT RESERVE OUT command with CLEAR service action (see 5.6.11.6); 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 5.6.11.3).

When a reservation key (i.e., registration) has been removed, no information shall be reported for that unregistered I_T nexus in subsequent READ KEYS service actions until the I_T nexus is registered again (see 5.6.7).

If the persist through power loss capability is not enabled, loss of power also causes persistent reservations to be released and registrations to be removed. When the most recent APTPL value received by the device server is zero (see 6.14.3), a power cycle:

- a) Releases all persistent reservations; and
- b) Removes all registered reservation keys (see 5.6.7).

5.6.11.2 Releasing

Only ~~a~~ the persistent reservation holder (see 5.6.10) is allowed to release a persistent reservation.

An application client releases the persistent reservation by issuing a PERSISTENT RESERVE OUT command with RELEASE service action through an I_T nexus that is a persistent reservation holder with the following parameters:

- a) RESERVATION KEY field set to the value of the reservation key that is registered with the logical unit for the I_T nexus; and
- b) TYPE field and SCOPE field set to match the persistent reservation being released.

In response to a persistent reservation release request from ~~a~~ the persistent reservation holder the device server shall perform a release by doing the following as an uninterrupted series of actions:

- a) Release the persistent reservation;
- b) Not remove any registration(s);
- c) If the released persistent reservation is a registrants only type or all registrants type persistent reservation, the device server shall establish a unit attention condition for the initiator port associated with every registered I_T nexus other than I_T nexus on which the PERSISTENT RESERVE OUT command with RELEASE service action was received, with the additional sense code set to RESERVATIONS RELEASED; ~~and~~
- d) If the released persistent reservation is a team type persistent reservation, the device server shall establish a unit attention condition for the initiator port associated with every reservation holder other than the I_T nexus on which the PERSISTENT RESERVE OUT command with RELEASE service action was received, with the additional sense code set to RESERVATIONS RELEASED; and
- e) If the persistent reservation is of any other type, the device server shall not establish a unit attention condition.

The established persistent reservation shall not be altered and the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID

RELEASE OF PERSISTENT RESERVATION, for a PERSISTENT RESERVE OUT command that specifies the release of a persistent reservation if:

- a) The requesting I_T nexus is a persistent reservation holder (see 5.6.10); and
- b) The SCOPE and TYPE fields do not match the scope and type of the established persistent reservation.

If there is no persistent reservation or in response to a persistent reservation release request from a registered I_T nexus that is not a persistent reservation holder (see 5.6.10), the device server shall do the following:

- a) Not release the persistent reservation, if any;
- b) Not remove any registrations; and
- c) Return GOOD status.

5.6.11.3 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 is the last remaining registered I_T nexus, then the device server shall also release the persistent reservation.~~

The device server shall also release the persistent reservation if:

- a) the I_T nexus is a reservation holder (see clause 5.6.10);
- b) the persistent reservation is of a type:
 - A) Write Exclusive – All Registrants;
 - B) Exclusive Access – All Registrants;
 - C) Write Exclusive – Team; or
 - D) Exclusive Access – Team;

and

- c) the I_T nexus is the last remaining reservation holder.

If the I_T nexus is the reservation holder and the persistent reservation is of a type other than all registrants or team reservation, then the device server shall also release the persistent reservation. If the persistent reservation is a registrants only type, the device server shall establish a unit attention condition for the initiator port associated with every registered I_T nexus except for the I_T nexus on which the PERSISTENT RESERVE OUT command was received, with the additional sense code set to RESERVATIONS RELEASED.

5.6.11.4 Preempting

5.6.11.4.1 Overview

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

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

Table 5 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 5 — Preempting actions

| Reservation Type | Service Action Reservation Key | Action | Reference |
|----------------------|--------------------------------------|---|----------------------------|
| All Registrants | Zero | Preempt the persistent reservation and remove registrations. | 5.6.11.4.3 |
| | Not Zero | Remove registrations. | 5.6.11.4.4 |
| Team | Zero | Preempt the persistent reservation and remove registrations. | 5.6.11.4.3 |
| | Not Zero | Remove registrations. | 5.6.11.4.4 |
| All other types | Zero | Terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. | |
| | Reservation holder's reservation key | Preempt the persistent reservation and remove registrations. | 5.6.11.4.3 |
| | Any other, non-zero reservation key | Remove registrations. | 5.6.11.4.4 |

See figure 1 for a description of how a device server interprets a PREEMPT service action to determine its actions (e.g., preempt the persistent reservation, remove registration, or both preempt the persistent reservation and remove registration).

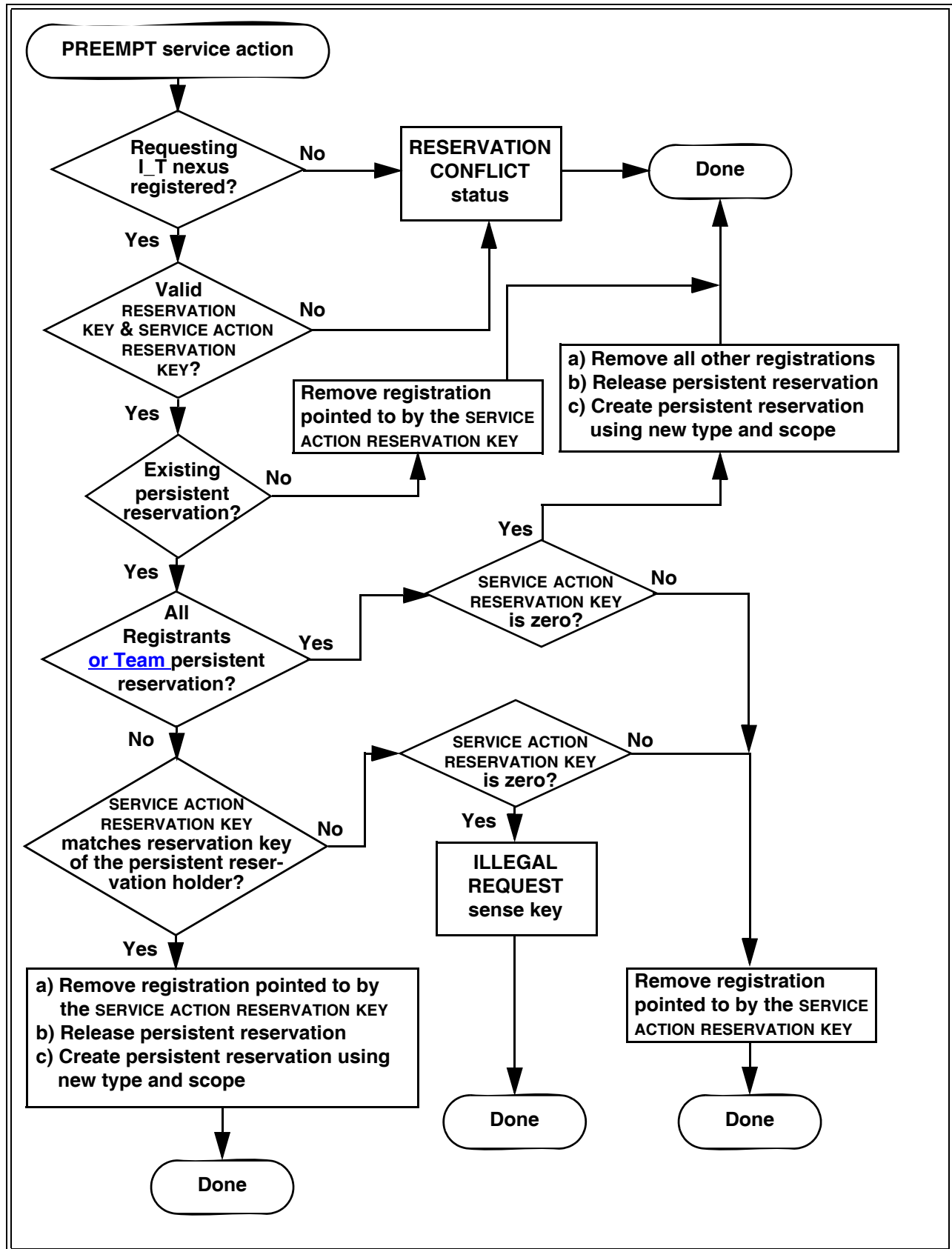


Figure 1 — Device server interpretation of PREEMPT service action

5.6.11.4.2 Failed persistent reservation preempt

Editor's Note 8: No changes to this section.

5.6.11.4.3 Preempting persistent reservations and registration handling

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

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

If the SERVICE ACTION RESERVATION KEY field identifies a persistent reservation holder (see 5.6.10), [then](#) the device server shall perform a preempt by doing the following as an uninterrupted series of actions:

- a) Release the persistent reservation for the holder identified by the SERVICE ACTION RESERVATION KEY field;
- b) Remove the registrations for all I_T nexuses identified by the SERVICE ACTION RESERVATION KEY field, except the I_T nexus that is being used for the PERSISTENT RESERVE OUT command. If an all registrants persistent reservation [or team persistent reservation](#) is present and the SERVICE ACTION RESERVATION KEY field is set to zero, then all registrations shall be removed except for that of the I_T nexus that is being used for the PERSISTENT RESERVE OUT command.
- c) Establish a persistent reservation for the preempting I_T nexus using the contents of the SCOPE and TYPE fields;
- d) Process tasks as defined in 5.6.1;
- e) Establish a unit attention condition for the initiator port associated with every I_T nexus that lost its persistent reservation and/or registration, with the additional sense code set to REGISTRATIONS PREEMPTED; and
- f) If the type or scope has changed, then for every I_T nexus whose reservation key was not removed, except for the I_T nexus on which the PERSISTENT RESERVE OUT command was received, the device server shall establish a unit attention condition for the initiator port associated with that I_T nexus, with the additional sense code set to RESERVATIONS RELEASED. If the type or scope have not changed, then no unit attention condition(s) shall be established for this reason.

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 enabled state (see SAM-4) 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 by a PERSISTENT RESERVE OUT command with the PREEMPT AND ABORT service action and TAS bit set to zero in the Control mode page (see 7.4.6).

If an all registrants or team persistent reservation is not present, it is not an error for the persistent reservation holder to preempt itself (i.e., 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 that is received from the persistent reservation holder). In that case, the device server shall establish the new persistent reservation and maintain the registration.

5.6.11.4.4 Removing registrations

Editor's Note 9: N.B. There was a lot of confusion in CAP on what rewording should be done in the following paragraph. I followed the spirit of what CAP said to do but took editorial license.

~~When a registered reservation key does not identify a persistent reservation holder (see 5.6.10), 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 nexus; and
- b) SERVICE ACTION RESERVATION KEY field set to match the reservation key of the registration or registrations being removed and the reservation key does not match the reservation key of the reservation holder (see clause 5.6.10).

Editor's Note 10: January CAP meeting reviewed to here. Plus suggestion to change team reservation key in the command section to a descriptor (see clause 6.14.4)

If the SERVICE ACTION RESERVATION KEY field does not identify a persistent reservation holder or there is no persistent reservation holder (i.e., there is no persistent reservation), then the device server shall perform a preempt by doing the following in an uninterrupted series of actions:

- a) Remove the registrations for all I_T nexuses specified by the SERVICE ACTION RESERVATION KEY field;
- 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 the initiator port associated with every I_T nexus that lost its registration other than the I_T nexus on which the PERSISTENT RESERVE OUT command was received, with the additional sense code 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, then 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, however, no unit attention condition is established for the I_T nexus on which the PERSISTENT RESERVE OUT command was received. The registration is removed.

5.6.11.5 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 5.6.11.4) except for the additions described in this subclause. If no reservation conflict occurred, the device server shall perform the following uninterrupted series of actions:

- a) If the persistent reservation is not an all registrants type [or a team type](#) then:
 - A) If the TST field is 000b (see 7.4.6) and the faulted I_T nexus (see 3.1.37), if any, is not the I_T nexus associated with the persistent reservation or registration being preempted, then the task set ACA condition shall be processed as defined in SAM-4;
 - B) If the TST field contains 000b and the faulted I_T nexus, if any, is the I_T nexus associated with the persistent reservation or registration being preempted, then the PERSISTENT RESERVE OUT command shall be processed without regard for the task set ACA condition; or
 - C) If the TST field contains 001b, then the ACA condition shall be processed as defined in SAM-4;
- b) Perform the uninterrupted series of actions described for the PREEMPT service action (see 5.6.11.4);
- c) All tasks from the I_T nexus(es) associated with the persistent reservations or registrations being preempted (i.e., preempted tasks) except the task containing the PERSISTENT RESERVE OUT command itself shall be aborted as defined in SAM-4. If an aborted task is a command that causes the device server to generate additional commands and data transfers (e.g., EXTENDED COPY), then all commands and data transfers generated by the command shall be aborted 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 I_T nexus;
- d) If the persistent reservation is not an all registrants type, then the device server shall clear any ACA condition associated with an I_T nexus being preempted and shall abort any tasks with an ACA attribute received on that I_T nexus;
- e) [If the persistent reservation is a team type, then:](#)
 - A) [If the SERVICE ACTION RESERVATION KEY is set to zero, then the device server shall do the following for any I_T nexus that is a persistent reservation holder:](#)
 - a) [Clear any ACA condition; and](#)
 - b) [Abort any tasks with an ACA attribute; or](#)
 - B) [If the SERVICE ACTION RESERVATION KEY is not set to zero, then the device server shall do the following for any I_T nexus registered using the specified reservation key:](#)
 - a) [Clear any ACA condition; and](#)
 - b) [Abort any tasks with an ACA attribute;](#)
- f) If the persistent reservation is an all registrants type, then:
 - A) If the service action reservation key is set to zero, the device server shall clear any ACA condition and shall abort any tasks with an ACA attribute; or
 - B) If the service action reservation key is not set to zero, the device server shall do the following for any I_T nexus registered using the specified reservation key:
 - a) Clear any ACA condition; and
 - b) Abort any tasks with an ACA attribute;
- and
- g) For logical units that implement the PREVENT ALLOW MEDIUM REMOVAL command (see SBC-3, SSC-3, and SMC-3), the device server shall perform an action equivalent to the processing of a PREVENT ALLOW MEDIUM REMOVAL command with the PREVENT field equal to zero received on the I_T nexuses associated with the persistent reservation being preempted.

The actions described in this subclause shall be performed for all I_T nexuses that are registered with the non-zero SERVICE ACTION RESERVATION KEY value, without regard for whether the preempted I_T nexuses hold the persistent reservation. If the SERVICE ACTION RESERVATION KEY value is zero and an all registrants persistent reservation is present, the device server shall abort all tasks for all registered I_T nexuses. [If the SERVICE ACTION RESERVATION KEY value is zero and a team persistent reservation is present, the device server shall abort all tasks for all I_T nexuses that were reservation holders.](#)

5.6.11.6 Clearing

[Editor's Note 11: No changes in this section.](#)

6.13 PERSISTENT RESERVE IN command

6.13.1 PERSISTENT RESERVE IN command introduction

[Editor's Note 12: No changes to this section.](#)

6.13.2 READ KEYS service action

[Editor's Note 13: No Changes to this section.](#)

6.13.3 READ RESERVATION service action

6.13.3.1 READ RESERVATION service action introduction

[Editor's Note 14: No changes in this section.](#)

6.13.3.2 Format of PERSISTENT RESERVE IN parameter data for READ RESERVATION

[Editor's Note 15: No changes in this section.](#)

6.13.3.3 Persistent reservations scope

[Editor's Note 16: No changes in this section.](#)

6.13.3.4 Persistent reservations type

The TYPE field (see table 6) specifies the characteristics of the persistent reservation being established for all logical blocks within the logical unit. Table 31 (see 5.6.1) defines the persistent reservation types under which each command defined in this standard is allowed to be processed. Each other command standard (see 3.1.17) defines the persistent reservation types under which each command defined in that command standard is allowed to be processed.

Table 6 — Persistent reservation TYPE field

| Code | Name | Description |
|-------------------------|---|---|
| 0h | | Obsolete |
| 1h | Write Exclusive | Access Restrictions: Some commands (e.g., media-access write commands) are only allowed for the persistent reservation holder (see 5.6.9). Persistent Reservation Holder: There is only one persistent reservation holder. |
| 2h | | Obsolete |
| 3h | Exclusive Access | Access Restrictions: Some commands (e.g., media-access commands) are only allowed for the persistent reservation holder (see 5.6.9). Persistent Reservation Holder: There is only one persistent reservation holder. |
| 4h | | Obsolete |
| 5h | Write Exclusive – Registrants Only | Access Restrictions: Some commands (e.g., media-access write commands) are only allowed for registered I_T nexuses. Persistent Reservation Holder: There is only one persistent reservation holder (see 5.6.9). |
| 6h | Exclusive Access – Registrants Only | Access Restrictions: Some commands (e.g., media-access commands) are only allowed for registered I_T nexuses. Persistent Reservation Holder: There is only one persistent reservation holder (see 5.6.9). |
| 7h | Write Exclusive – All Registrants | Access Restrictions: Some commands (e.g., media-access write commands) are only allowed for registered I_T nexuses. Persistent Reservation Holder: Each registered I_T nexus is a persistent reservation holder (see 5.6.9). |
| 8h | Exclusive Access – All Registrants | Access Restrictions: Some commands (e.g., media-access commands) are only allowed for registered I_T nexuses. Persistent Reservation Holder: Each registered I_T nexus is a persistent reservation holder (see 5.6.9). |
| 9h | Write Exclusive – Team | Access Restrictions: Some commands (e.g., media-access write commands) are only allowed for a persistent reservation holder (see 5.6.9). Persistent Reservation Holder: Each I_T nexus that is part of the team reservation. |
| Ah | Exclusive Access – Team | Access Restrictions: Some commands (e.g., media-access write commands) are only allowed for a persistent reservation holder (see 5.6.9). Persistent Reservation Holder: Each I_T nexus that is part of the team reservation |
| Bh - Fh | Reserved | |

6.13.4 REPORT CAPABILITIES service action

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

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the REPORT CAPABILITIES service action is shown in table 7.

Table 7 — PERSISTENT RESERVE IN parameter data for REPORT CAPABILITIES

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

The LENGTH field indicates the length in bytes of the parameter data. The relationship between the LENGTH field and the CDB ALLOCATION LENGTH field is defined in 4.3.4.6.

A Compatible Reservation Handling (CRH) bit set to one indicates that the device server supports the exceptions to the SPC-2 RESERVE and RELEASE commands described in 5.6.3. A CRH bit set to zero indicates that RESERVE(6) command, RESERVE(10) command, RELEASE(6) command, and RELEASE(10) command are processed as defined in SPC-2.

A Specify Initiator Ports Capable (SIP_C) bit set to one indicates that the device server supports the SPEC_I_PT bit in the PERSISTENT RESERVE OUT command parameter data (see 6.14.3). An SIP_C bit set to zero indicates that the device server does not support the SPEC_I_PT bit in the PERSISTENT RESERVE OUT command parameter data.

An All Target Ports Capable (ATP_C) bit set to one indicates that the device server supports the ALL_TG_PT bit in the PERSISTENT RESERVE OUT command parameter data. An ATP_C bit set to zero indicates that the device server does not support the ALL_TG_PT bit in the PERSISTENT RESERVE OUT command parameter data.

A Persist Through Power Loss Capable (PTPL_C) bit set to one indicates that the device server supports the persist through power loss capability (see 5.6.4) for persistent reservations and the APTPL bit in the PERSISTENT RESERVE OUT command parameter data. An PTPL_C bit set to zero indicates that the device server does not support the persist through power loss capability.

A Type Mask Valid (TMV) bit set to one indicates that the PERSISTENT RESERVATION TYPE MASK field contains a bit map indicating which persistent reservation types are supported by the device server. A TMV bit set to zero indicates that the PERSISTENT RESERVATION TYPE MASK field shall be ignored.

The ALLOW COMMANDS field (see table 8) indicates whether certain commands are allowed through certain types of persistent reservations.

Table 8 — ALLOW COMMANDS field

| Code | Description |
|-------------|---|
| 000b | No information is provided about whether certain commands are allowed through certain types of persistent reservations. |
| 001b | The device server allows the TEST UNIT READY command (see table 31 in 5.6.1) through Write Exclusive and Exclusive Access persistent reservations and does not provide information about whether the following commands are allowed through Write Exclusive persistent reservations: a) the MODE SENSE, READ ATTRIBUTE, READ BUFFER, RECEIVE DIAGNOSTIC RESULTS, REPORT SUPPORTED OPERATION CODES, and REPORT SUPPORTED TASK MANAGEMENT FUNCTION commands (see table 31 in 5.6.1); and b) the READ DEFECT DATA command (see SBC-3). |
| 010b | The device server allows the TEST UNIT READY command through Write Exclusive and Exclusive Access persistent reservations and does not allow the following commands through Write Exclusive persistent reservations: a) the MODE SENSE, READ ATTRIBUTE, READ BUFFER, RECEIVE DIAGNOSTIC RESULTS, REPORT SUPPORTED OPERATION CODES, and REPORT SUPPORTED TASK MANAGEMENT FUNCTION commands; and b) the READ DEFECT DATA command. |
| 011b | The device server allows the TEST UNIT READY command through Write Exclusive and Exclusive Access persistent reservations and allows the following commands through Write Exclusive persistent reservations: a) the MODE SENSE, READ ATTRIBUTE, READ BUFFER, RECEIVE DIAGNOSTIC RESULTS, REPORT SUPPORTED OPERATION CODES, and REPORT SUPPORTED TASK MANAGEMENT FUNCTION commands; and b) the READ DEFECT DATA command. |
| 100b - 111b | Reserved |

[A Team Member List Capable \(TML_C\) bit set to one indicates that the device server supports receipt of a member list to set members of a team reservation \(see 5.6.2\). A TML_C bit set to zero indicates the device server does not support receipt of a member list to set members of a team reservation.](#)

A Persist Through Power Loss Activated (PTPL_A) bit set to one indicates that the persist through power loss capability is activated (see 5.6.4). A PTPL_A bit set to zero indicates that the persist through power loss capability is not activated.

The PERSISTENT RESERVATION TYPE MASK field (see table 9) contains a bit map that indicates the persistent reservation types that are supported by the device server.

Table 9 — Persistent Reservation Type Mask format

| Bit Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|----------|----------|----------|----------|----------|-------|-------------------------|-------------------------|----------|
| 4 | WR_EX_AR | EX_AC_RO | WR_EX_RO | Reserved | EX_AC | Reserved | WR_EX | Reserved |
| 5 | Reserved | | | | | EX_AC_I | WR_EX_I | EX_AC_AR |

A Write Exclusive – All Registrants (WR_EX_AR) bit set to one indicates that the device server supports the Write Exclusive – All Registrants persistent reservation type. An WR_EX_AR bit set to zero indicates that the device server does not support the Write Exclusive – All Registrants persistent reservation type.

An Exclusive Access – Registrants Only (EX_AC_RO) bit set to one indicates that the device server supports the Exclusive Access – Registrants Only persistent reservation type. An EX_AC_RO bit set to zero indicates that the device server does not support the Exclusive Access – Registrants Only persistent reservation type.

A Write Exclusive – Registrants Only (WR_EX_RO) bit set to one indicates that the device server supports the Write Exclusive – Registrants Only persistent reservation type. An WR_EX_RO bit set to zero indicates that the device server does not support the Write Exclusive – Registrants Only persistent reservation type.

An Exclusive Access (EX_AC) bit set to one indicates that the device server supports the Exclusive Access persistent reservation type. An EX_AC bit set to zero indicates that the device server does not support the Exclusive Access persistent reservation type.

A Write Exclusive (WR_EX) bit set to one indicates that the device server supports the Write Exclusive persistent reservation type. An WR_EX bit set to zero indicates that the device server does not support the Write Exclusive persistent reservation type.

[An Exclusive Access – Team \(EX_AC_T\) bit set to one indicates that the device server supports the Exclusive Access – Team persistent reservation type. An EX_AC_T bit set to zero indicates that the device server does not support the Exclusive Access – Team persistent reservation type.](#)

[A Write Exclusive – Team \(WR_EX_T\) bit set to one indicates that the device server supports the Write Exclusive – Team persistent reservation type. An WR_EX_T bit set to zero indicates that the device server does not support the Write Exclusive – Team persistent reservation type.](#)

An Exclusive Access – All Registrants (EX_AC_AR) bit set to one indicates that the device server supports the Exclusive Access – All Registrants persistent reservation type. An EX_AC_AR bit set to zero indicates that the device server does not support the Exclusive Access – All Registrants persistent reservation type.

6.13.5 READ FULL STATUS service action

[Editor's Note 17: No changes to this section.](#)

6.14 PERSISTENT RESERVE OUT command

6.14.1 PERSISTENT RESERVE OUT command introduction

The PERSISTENT RESERVE OUT command (see table 10) is used to request service actions that reserve a logical unit 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 10 — PERSISTENT RESERVE OUT command

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

If a PERSISTENT RESERVE OUT command is attempted, but there are insufficient device server resources to complete the operation, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code 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 SCOPE field and TYPE field are defined in 6.13.3.3 and 6.13.3.4. If a SCOPE field specifies a scope that is not implemented, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code 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.

The PARAMETER LIST LENGTH field specifies the number of bytes of parameter data for the PERSISTENT RESERVE OUT command.

The parameter list shall be 24 bytes in length and the PARAMETER LIST LENGTH field shall contain 24 (18h), if the following conditions are true:

- a) The SPEC_I_PT bit (see 6.14.3) is set to zero; and
- b) The service action is not REGISTER AND MOVE.

If the SPEC_I_PT bit is set to zero, the service action is not REGISTER AND MOVE, and the parameter list length is not 24, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

If the parameter list length is larger than the device server is able to process, the command should be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to PARAMETER LIST LENGTH ERROR.

6.14.2 PERSISTENT RESERVE OUT service actions

When processing the PERSISTENT RESERVE OUT service actions, the device server shall increment the PRgeneration value as specified in 6.13.2.

The PERSISTENT RESERVE OUT command service actions are defined in table 11.

Table 11 — PERSISTENT RESERVE OUT service action codes

| Code | Name | Description | PRGENERATION field incremented (see 6.13.2) | Parameter list format |
|-----------|----------------------------------|--|---|--|
| 00h | REGISTER | Register a reservation key with the device server (see 5.6.6) or unregister a reservation key (see 5.6.10.3). | Yes | Basic (see 6.14.3) |
| 01h | RESERVE | Creates a persistent reservation having a specified SCOPE and TYPE (see 5.6.8). The SCOPE and TYPE of a persistent reservation are defined in 6.13.3.3 and 6.13.3.4. | No | Basic (see 6.14.3) Reservation (see 6.14.4) |
| 02h | RELEASE | Releases the selected persistent reservation (see 5.6.10.2). | No | Basic (see 6.14.3) |
| 03h | CLEAR | Clears all reservation keys (i.e., registrations) and all persistent reservations (see 5.6.10.6). | Yes | Basic (see 6.14.3) |
| 04h | PREEMPT | Preempts persistent reservations and/or removes registrations (see 5.6.10.4). | Yes | Basic (see 6.14.3) Reservation (see 6.14.4) |
| 05h | PREEMPT AND ABORT | Preempts persistent reservations and/or removes registrations and aborts all tasks for all preempted I_T nexuses (see 5.6.10.4 and 5.6.10.5). | Yes | Basic (see 6.14.3) Reservation (see 6.14.4) |
| 06h | REGISTER AND IGNORE EXISTING KEY | Register a reservation key with the device server (see 5.6.6) or unregister a reservation key (see 5.6.10.3). | Yes | Basic (see 6.14.3) |
| 07h | REGISTER AND MOVE | Register a reservation key for another I_T nexus with the device server and move a persistent reservation to that I_T nexus (see 5.6.7) | Yes | Register and move (see 6.14.5) |
| 08h - 1Fh | Reserved | | | |

6.14.3 Basic PERSISTENT RESERVE OUT parameter list

Editor's Note 18: Changes to this section are only intended to remove the references to the RESERVE, PREEMPT, and PREEMPT AND ABORT service actions which are covered in the new clause 6.14.4.

The parameter list format shown in table 12 shall be used by the PERSISTENT RESERVE OUT command with ~~any~~:

- a) [the REGISTER service action;](#)
- b) [the RELEASE service action;](#)
- c) [the CLEAR service action; and](#)
- d) [the REGISTER AND IGNORE EXISTING KEY service action.](#)

~~service action except the REGISTER AND MOVE service action.~~ All fields shall be sent, even if the field is not required for the specified service action and scope values.

Table 12 — PERSISTENT RESERVE OUT parameter list

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

The obsolete fields in bytes 16 through 19, byte 22 and byte 23 were defined in a previous standard.

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 command 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 the following service actions: REGISTER, [and REGISTER AND IGNORE EXISTING KEY](#), ~~PREEMPT, and PREEMPT AND ABORT~~. The SERVICE ACTION RESERVATION KEY field is ignored for the following service actions: RESERVE, RELEASE, and CLEAR.

For the REGISTER service action 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 service action and PREEMPT AND ABORT service action, 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 5.6.9), persistent reservations that are to be preempted.~~

If the Specify Initiator Ports (SPEC_I_PT) bit is set to zero, the device server 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 AND IGNORE EXISTING KEY service action, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST. If the SPEC_I_PT bit is set to one for the REGISTER service action, the additional parameter data (see table 16) shall include a list of transport IDs and the device server shall also 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 logical unit does not have enough resources available to hold the registration information), none of the other registrations shall be made.

Table 13 — PERSISTENT RESERVE OUT specify initiator ports additional parameter data

| Bit Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------|--|---|---|---|---|---|---|---|
| 24 | TRANSPORTID PARAMETER DATA LENGTH (n - 27) | | | | | | | |
| 27 | | | | | | | | |
| | TransportIDs list | | | | | | | |
| 28 | First TransportID | | | | | | | |
| | ⋮ | | | | | | | |
| | Last TransportID | | | | | | | |
| n | | | | | | | | |

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

The command shall be terminated with CHECK CONDITION status, with 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 Target Ports (ALL_TG_PT) bit is valid only for the REGISTER service action and the REGISTER AND IGNORE EXISTING KEY service action, 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 service action or a 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 SCSI target device known to the device server (i.e., as if the same registration request had been received individually through each target port). If the device server receives a REGISTER service action or a 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. If a device server that does not support an ALL_TG_PT bit set to one receives that value in a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The Activate Persist Through Power Loss (APTPL) bit is valid only for the REGISTER service action and the REGISTER AND IGNORE EXISTING KEY service action, 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 service action or a REGISTER AND IGNORE EXISTING KEY service action, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code 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 5.6.6). 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 5.6.4).

Table 17 summarizes which fields are set by the application client and interpreted by the device server for each service action and scope value.

Table 14 — PERSISTENT RESERVE OUT service actions and valid parameters (part 1 of 2)

| Service action | Allowed SCOPE | Parameters (part 1 of 2) | | | |
|----------------------------------|---------------------|--------------------------|------------------|--------------------------------|-----------------------------|
| | | TYPE | RESERVATION KEY | SERVICE ACTION RESERVATION KEY | APTPL |
| REGISTER | ignored | ignored | valid | valid | valid |
| REGISTER AND IGNORE EXISTING KEY | ignored | ignored | ignored | valid | valid |
| RESERVE | LU_SCOPE | valid | valid | ignored | ignored |
| RELEASE | LU_SCOPE | valid | valid | ignored | ignored |
| CLEAR | ignored | ignored | valid | ignored | ignored |
| PREEMPT | LU_SCOPE | valid | valid | valid | ignored |
| PREEMPT AND ABORT | LU_SCOPE | valid | valid | valid | ignored |
| REGISTER AND MOVE | LU_SCOPE | valid | valid | valid | not applicable ^a |

^a The parameter list format for the REGISTER AND MOVE service action is described in 6.14.5.
 NOTE 1 The parameter list format and valid parameters for the RESERVE, PREEMPT, and PREEMPT AND ABORT service actions is described in 6.14.4

Table 17 — PERSISTENT RESERVE OUT service actions and valid parameters (part 2 of 2)

| Service action | Allowed SCOPE | Parameters (part 2 of 2) | |
|----------------------------------|---------------------|-----------------------------|-----------------------------|
| | | ALL_TG_PT | SPEC_I_PT |
| REGISTER | ignored | valid | valid |
| REGISTER AND IGNORE EXISTING KEY | ignored | valid | valid |
| RESERVE | LU_SCOPE | ignored | ignored |
| RELEASE | LU_SCOPE | ignored | ignored |
| CLEAR | ignored | ignored | ignored |
| PREEMPT | LU_SCOPE | ignored | ignored |
| PREEMPT AND ABORT | LU_SCOPE | ignored | ignored |
| REGISTER AND MOVE | LU_SCOPE | not applicable ^a | not applicable ^a |

^a The parameter list format for the REGISTER AND MOVE service action is described in 6.14.5.
 NOTE 1 The parameter list format and valid parameters for the RESERVE, PREEMPT, and PREEMPT AND ABORT service actions is described in 6.14.4

6.14.4 Reservation PERSISTENT RESERVE OUT parameter list

Editor’s Note 19: This entire section is new. However it is a copy of the BASIC parameter list with changes to accomodate team reservations. Text that is unchanged from BASIC parameter list is left black so we can watch for backward compatibility.

The parameter list format shown in table 15 shall be used by the PERSISTENT RESERVE OUT command with:

- a) [the RESERVE service action:](#)
- b) [the PREEMPT service action: and](#)
- c) [the PREEMPT AND ABORT service action.](#)

All fields [through byte 23](#) shall be sent, even if the field is not required for the specified service action and scope values. [If the TYPE field is set to Write Exclusive – Team or Exclusive Access – Team, then the TEAM RESERVATION KEY field shall also be sent.](#)

Table 15 — Reservation PERSISTENT RESERVE OUT parameter list

| Bit Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|-------------|--------------------------------------|---|---|-----------|-----------|----------|-------|-------|
| 0 | (MSB) | | | | | | | |
| 7 | RESERVATION KEY | | | | | | | (LSB) |
| 8 | (MSB) | | | | | | | |
| 15 | SERVICE ACTION RESERVATION KEY | | | | | | | (LSB) |
| 16 | Obsolete | | | | | | | |
| 19 | Obsolete | | | | | | | |
| 20 | Reserved | | | SPEC_I_PT | ALL_TG_PT | Reserved | APTPL | |
| 21 | Reserved | | | | | | | |
| 22 | Obsolete | | | | | | | |
| 23 | Obsolete | | | | | | | |
| 24 | (MSB) | | | | | | | |
| 27 | TEAM RESERVATION KEY | | | | | | | (LSB) |
| 28 | Additional parameter data | | | | | | | |
| n | | | | | | | | |

The obsolete fields in bytes 16 through 19, byte 22 and byte 23 were defined in a previous standard.

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 command 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~~ [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~~ [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 the following service actions: ~~REGISTER, REGISTER AND IGNORE EXISTING KEY, PREEMPT, and PREEMPT AND ABORT.~~ The SERVICE ACTION RESERVATION KEY field is ignored for the ~~following service actions: RESERVE, RELEASE, and CLEAR service action.~~

~~For the REGISTER service action 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 service action and PREEMPT AND ABORT service action, 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 5.6.9), persistent reservations that are to be preempted.

If the TYPE field is set to Write Exclusive – Team or Exclusive Access – Team, then the TEAM RESERVATION KEY field (see table 13) is valid and the additional parameter data may contain PERSISTENT RESERVE OUT specify initiator ports additional parameter data (see table 16).

If the Specify Initiator Ports (SPEC_I_PT) bit is set to zero, the device server shall apply the registration reservation 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 AND IGNORE EXISTING KEY service action, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.~~ If the SPEC_I_PT bit is set to one and the TYPE field is set to Write Exclusive – Team or Exclusive Access – Team for the REGISTER service action, the additional parameter data (see table 16) shall include a list of transport IDs and the device server shall also apply the registration team reservation to the I_T nexus for each initiator port specified by a TransportID. If a registration reservation fails for any initiator port (e.g., if the logical unit does not have enough resources available to hold the registration team reservation information), none of the other registrations team reservations shall be made. If the SPEC_I_PT bit is set to one and the TYPE field is not set to Write Exclusive – Team or Exclusive Access – Team, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Editor's Note 20: What does, "...non of the other registrations shall be made." mean? Does it mean: 1) Only the I_T nexus on which the command received gets a registration; 2) All I_T nexus prior to running out of resources gets a registration; or 3) Nobody gets a registration? Whatever it is for Registration should probably be the same for reservation to maintain consistency.

If the I_T nexus for an initiator port specified by a TransportID is not registered, then the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

Editor's Note 21: I had intended to allow the I_T nexus sending the command to not be included in the list. Some in CAP had an issue with this but others liked it. To further add to the discussion, does the member list need to contain the I_T nexus on which the command is received? Stated differently, since the team reservation key is required to establish the reservation and the member list is an optional add on, does the member list only apply to additional I_T nexuses. I think that is probably cleaner - in which case this question is answered.

It shall not be considered an error if a team reservation is in effect and a PERSISTENT RESERVE OUT command is received with the SPEC I PT bit is set to one for the RESERVE service action with the TYPE field set to Write Exclusive – Team or Exclusive Access – Team with the additional parameter data (see table 16) containing a list of transport IDs that does not match the list of reservation holders. In this case, a new team reservation is created with the new list of transport IDs. The device server shall establish a unit attention condition for the initiator port associated with every reservation holder that loses their reservation except for the I T nexus on which the PERSISTENT RESERVE OUT command was received, with the additional sense code set to RESERVATIONS RELEASED.

Table 16 — PERSISTENT RESERVE OUT specify initiator ports additional parameter data

| Bit Byte | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---------------------|---|---|---|---|---|---|---|---|
| <u>24</u> <u>28</u> | TRANSPORTID PARAMETER DATA LENGTH (n - <u>1</u> <u>27</u>) | | | | | | | |
| <u>27</u> <u>31</u> | | | | | | | | |
| | TransportIDs list | | | | | | | |
| <u>28</u> <u>32</u> | First TransportID | | | | | | | |
| | ⋮ | | | | | | | |
| n | Last TransportID | | | | | | | |

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

The command shall be terminated with CHECK CONDITION status, with 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 Target Ports (ALL_TG_PT) bit is valid only for the REGISTER service action and the REGISTER AND IGNORE EXISTING KEY service action, 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 service action or a 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 SCSI target device known to the device server (i.e., as if the same registration request had been received individually through each target port). If the device server receives a REGISTER service action or a 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. If a device server that does not support an ALL_TG_PT bit set to one receives that value in a REGISTER service action or a REGISTER AND IGNORE EXISTING KEY service action, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.~~

Support for the ALL_TG_PT bit is optional.

If the device server receives a RESERVE, PREEMPT, or PREEMPT AND ABORT service action with the TYPE field set to Write Exclusive – Team or Exclusive Access – Team and with the ALL_TG_PT BIT set to one, then it shall create the specified reservation on all target ports in the SCSI target device known to the device server (i.e., as if

the same reservation request had been received individually through each target port). If the device server receives a RESERVE, PREEMPT, or PREEMPT AND ABORT service action with the TYPE field set to Write Exclusive – Team or Exclusive Access – Team and with the ALL_TG_PT BIT set to zero, it shall apply the reservation only to the target port through which the PERSISTENT RESERVE OUT command was received. If a device server that does not support an ALL_TG_PT BIT set to one receives that value in a RESERVE, PREEMPT, or PREEMPT AND ABORT service action with the TYPE field set to Write Exclusive – Team or Exclusive Access – Team, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN PARAMETER LIST.

The Activate Persist Through Power Loss (APTPL) bit is valid only for the REGISTER service action and the REGISTER AND IGNORE EXISTING KEY service action, 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 service action or a REGISTER AND IGNORE EXISTING KEY service action, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code 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 5.6.6). 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 5.6.4).

Table 17 summarizes which fields are set by the application client and interpreted by the device server for each service action and scope value.

Table 17 — Reservation PERSISTENT RESERVE OUT service actions and valid parameters (part 1 of 2)

| Service action | Allowed SCOPE | Parameters (part 1 of 2) | | | |
|-------------------|---------------|--------------------------|-----------------|--------------------------------|---------|
| | | TYPE | RESERVATION KEY | SERVICE ACTION RESERVATION KEY | APTPL |
| RESERVE | LU_SCOPE | valid | valid | ignored | ignored |
| PREEMPT | LU_SCOPE | valid | valid | valid | ignored |
| PREEMPT AND ABORT | LU_SCOPE | valid | valid | valid | ignored |
| a | | | | | |

Table 17 — Reservation PERSISTENT RESERVE OUT service actions and valid parameters (part 2 of 2)

| Service action | Allowed SCOPE | TYPE | Parameters (part 2 of 2) | | |
|-----------------------------------|---------------|---|--------------------------|-----------------------|-------------------------|
| | | | ALL_TG_PT | SPEC_I_PT | TEAM RESERVATION KEY |
| RESERVE | LU_SCOPE | Write Exclusive – Team or Exclusive Access – Team | valid | valid | valid |
| | | all others | ignored | ignored | ignored |
| PREEMPT | LU_SCOPE | Write Exclusive – Team or Exclusive Access – Team | valid | valid | valid |
| | | all others | ignored | ignored | ignored |
| PREEMPT AND ABORT | LU_SCOPE | Write Exclusive – Team or Exclusive Access – Team | valid | valid | valid |
| | | all others | ignored | ignored | ignored |
| a | | | | | |

6.14.5 PERSISTENT RESERVE OUT command with REGISTER AND MOVE service action parameters

[Editor’s Note 22: No change to this section.](#)
