Date: Sept 22, 1998

To: T10 Committee (SCSI)

From: George Penokie (IBM)

Subject: Persistent Reserations

5.3 Reservations

Commands that establish reservations may be used to restrict the execution of commands to a logical unit or a portion of the logical unit. Using the reservation commands, application clients may procure assistance from the device server to share and protect data or resources. If the application clients do not cooperate in the execution of a reservation protocol, data may be unexpectedly modified and deadlock conditions may occur.

This clause provides a general overview of reservations. The general description of reservations involves two groups of considerations; a) the type of reservation established, and b) the method used to establish, rescind, and manage the reservation. There are limits on the combinations of reservation types available under some reservation management methods. See the reservations management commands descriptions for details.

The types of reservations that can be established are:

a) **logical unit reservations** - a logical unit reservation restricts access to the entire logical unit; and
b) **element reservations** - an element reservation restricts access to a specified element within a medium changer.

The types of reservations can be further qualified by restrictions on types of access (e.g., read, write, control, etc.). However, access type restrictions are handled as an aspect of reservation management, not as an aspect of the type of reservation being established. In addition, some methods of reservations management permit establishing reservations on behalf of another device in the same SCSI domain (third-party reservations).

The methods of managing reservations are identified by the commands associated with the methods. The methods of managing reservations are:

a) Reserve/Release - associated with the RESERVE(10), RELEASE(10), RESERVE(6), and RELEASE(6) commands (see 7.21, 7.17, 7.22, and 7.18, respectively); and
b) Persistent Reservations - associated with the PERSISTENT RESERVE OUT and PERSISTENT RESERVE IN commands (see 7.13 and 7.12, respectively).

The reservation restrictions placed on commands that explicitly read or write the medium result from the type of reservation combined with any access qualifications. The details of the reading and writing restrictions are described in this standard in the clauses that define the commands associated with each management method. For the Reserve/Release management method, see 7.21. For the Persistent Reservations management method, see 7.13.

Commands that do not explicitly read or write the medium may conflict with the restrictions established by a previous reservation. If a reservation conflict precludes the execution of any part of the command, none of the command shall be performed and the device server shall terminate that command with a RESERVATION CONFLICT status. The command may be checked for reservation conflicts at any time between the time the command is delivered to the device server and the time the task representing the command first enters the enabled task state. The particular reservation restrictions imposed are highly dependent on the relationship between the command and the type of reservations. The restrictions do not depend on the reservation management method

and may not be the same as the restrictions for commands that explicitly read or write the medium. The reservation restrictions for commands that do not explicitly access the medium are defined in the device model clause or in the clause defining that specific command.

The reservation restrictions for commands that manage reservations may conflict with previous reservations. The clauses describing the reservations management commands define the interactions between the commands and previously established reservations. (See 7.12, 7.13, 7.17, 7.18, 7.21, and 7.22.)

Commands that explicitly read or write the medium may conflict with the restrictions established by a previous reservation. If a reservation conflict precludes the execution of any part of the command, none of the command shall be performed and the device server shall terminate that command with a RESERVATION CONFLICT status. The command may be checked for reservation conflicts at any time between the time the command is delivered to the device server and the time the device server begins to modify the data on the medium or deliver data from the medium. For each command, this standard or a related command standard (see 3.1.11) defines the conditions that result in RESERVATION CONFLICT. The conditions are defined in the device model clause or in the clause defining that specific command.

The time at which a reservation becomes active 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 active. An active reservation may apply to some or all tasks queued 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. The execution of a reservation command shall be performed as a single event, such that no other command processed by the device server shall be uncertain of the presence of a reservation at the time the command tests for the presence of a reservation.

Because a device server is unable to differentiate among the reservations made by different application clients running on an initiator, all application clients on the initiator have the same access restrictions. When multiple application clients are accessing a single device server from one initiator, the application clients should coordinate reservations and persistent reservations.

5.3.1 The Reserve/Release management method

The Reserve/Release management method commands, RESERVE(6), RESERVE(10), RELEASE(6), and RELEASE(10) are used among multiple initiators that do not require operations to be protected across initiator failures (and subsequent hard resets). The Reserve/Release reservations management method also allows an application client to provide restricted device access to one additional initiator (a third-party initiator), usually a temporary initiator performing a service for the application client sending the reservation command.

Reservations managed using the Reserve/Release method do not persist across some recovery actions (e.g., hard resets), so most systems require significant reinitialization after a failure that results in a hard reset. Reserve/Release managed reservations are retained by the device server until released or until reset by mechanisms specified in this standard.

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

5.3.2 The Persistent Reservations management method

The Persistent Reservations management method is used among multiple initiators that require operations to be protected across initiator failures, which usually involve hard resets. Even though different protocols that transport SCSI handle hard resets differently (e.g., parallel uses a reset signal, fibre channel uses log outs) the persistent reservation shall be protected. Persistent reservations persist across recovery actions, to provide initiators with more detailed control over reservations recovery. Persistent reservations for failing initiators may be preempted by another initiator as part of the recovery process. Persistent reservations shall be retained by the device server until

released, preempted, or until cleared by mechanisms specified in this standard. Persistent reservations are optionally retained when power to the target is lost.

Since persistent reservations are not reset by the TARGET RESET task management function or other global actions and may, optionally, be preserved across power cycles, they may be used to enforce device sharing among multiple initiators.

The PERSISTENT RESERVE OUT and PERSISTENT RESERVE IN commands provide the basic mechanism for dynamic contention resolution in multiple initiator systems using multiple port targets. The identification of persistent reservations using the reservation key (see clause 5.3.2.2.1) makes it possible to determine which initiator ~~port~~ holds the ~~conflicting~~ persistent reservation and to take over a persistent reservation from a failing or uncooperative initiator.

Before an initiator can establish a persistent reservation, the initiator must register with the device server. Registration has two purposes:

a) It defines a set of initiators that are participating in the persistent reservation management method. These initiators have common access privileges under certain reservation types; and
b) It allows the application client to establish an alias, known as a reservation key, for the combined initiator identifier and device server port identifier through which the registration request is received.

The reservation key provides a device-independent method to identify an initiator on a specific port of a device server. The reservation key is used to in the PERSISTENT RESERVE IN command to identify which initiators are registered, and which initiator holds the reservation. The reservation key is used in the PERSISTENT RESERVE OUT command to specify which initiator's registration or persistent reservation to preempt.

Reservation key values may be used by application clients to identify initiators, using application specific methods that are outside the scope of this standard. This standard provides the ability to register no more than one key per initiator/logical unit pair and multiple initiators may use the same key. This provides the ability for an application client to preempt multiple initiators with a single PERSISTENT RESERVE OUT command, but it removes the ability for the application client to uniquely identify the initiators using the PERSISTENT RESERVE commands.

The capability of preserving persistent reservations and registration keys across power cycles requires the use of a nonvolatile memory within the SCSI device. Any SCSI device that supports the Persist Through Power Loss (APRPL) capability of persistent reservation and has nonvolatile memory that is not ready shall accept the following commands:

a) Inquiry
b) Log Sense
c) Report LUNs
d) Request Sense
e) Start/Stop Unit
f) Test Unit Ready

Any commands, other than those listed above shall return CHECK CONDITION status. The sense key shall be set to NOT READY and the additional sense data shall be set as described in the TEST UNIT READY command (see 7.24).

5.3.2.1 Preserving persistent reservations across power cycles

The application client may request the device server to preserve the persistent reservation and registration keys across power cycles by requesting the Activate Persist Through Power Loss (APTPL) capability. The application client may request this as part of registration by setting the APTPL bit to one.

After the application client enables the APTPL capability the device server shall preserve all current and further registrations and persistent reservations associated with the logical unit that the REGISTER service action was addressed to until an application client disables the APTPL capability. The most recently received valid APTPL value from any application client shall govern the logical unit's behavior in the event of power loss.

When required the device server shall, at a minimum, preserve the following information for each registration:

a) Initiator identifier;
b) reservation key; and
c) when supported by the protocol, the initiator port's world wide identification.

When required the device server shall, at a minimum, preserve the following information for the reservation:

a) Initiator identifier;
b) reservation key;
c) scope;
d) type; and
e) when supported by the protocol, the initiator port's world wide identification.

When available, the initiator port's world wide identification shall be used to determine if the initiator identifier has changed. This determination shall be made at any time the target detects that the configuration of the system may have changed. If the initiator identification changed, the device server shall assign the new initiator identification to the existing registration and reservation.

5.3.2.2 Finding previous persistent reservations and reservation keys

The application client can obtain information about the persistent reservation and the reservation keys that are active within a device server by issuing PERSISTENT RESERVE IN commands with a READ KEY service action or a READ RESERVATION service action.

5.3.2.2.1 Reservation keys

To determine if any initiators have registered with a device server the application client shall:

a) Issue a PERSISTENT RESERVE IN command with a service action of READ KEYS.

In response to a PERSISTENT RESERVE IN with a READ KEYS service action the device server shall report the following: ~~if there are any registered initiators:~~

a) the current generation value (see clause 7.11.2); and
b) the reservation key for every initiator that is currently registered.

The generation value allows the application client examining the generation value to verify that the configuration of the initiators attached to a logical unit has not been modified by another application client without the knowledge of the examining application client.

A reservation key is the registered reservation key under which the reservation is held. The relationship between a reservation key and the initiator or port is outside the scope of this standard.

Each reservation key may be examined by the application client and correlated with a set of initiators and SCSI ports by mechanisms outside the scope of this standard. Duplicate keys are possible, if multiple initiators use the same reservation key.

An initiator may establish registrations for multiple logical units under a single SCSI device using any combination of unique or duplicate keys.

5.3.2.2.2 Reporting Persistent reservations

The PERSISTENT RESERVE IN command may be used to determine if any persistent reservation exists and to determine the scope, type, and reservation key for that persistent reservation.

Multiple persistent reservations may be created in SCSI devices that contain multiple elements. However, there shall only be one persistent reservation per element.

To receive the persistent reservation information an application client shall:

a) Issue a PERSISTENT RESERVE IN command with a service action of READ RESERVATION.

In response PERSISTENT RESERVE IN with a READ RESERVATION service action the device server shall report the following: ~~if a persistent reservation is active~~:

a) the current generation value (see clause 7.11.2);
b) if any, the registered reservation key associated with the initiator that holds the active persistent reservation;
c) if any, the scope and type of each active persistent reservation; and
d) if any, the scope-specific address.

An application client uses unique reservation keys for each initiator/logical unit pair it may use the reservation key to associate the persistent reservation with the initiator that holds the persistent reservation. This association is done using techniques that are outside the scope of this standard.

5.3.2.3 Registering

To establish a persistent reservation the initiator shall first register with a device server. This is accomplished by:

a) issuing a PERSISTENT RESERVE OUT command with service action of REGISTER;
b) optionally setting the APTPL bit; and
c) setting a non-zero reservation key.

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

NOTE 1 It is recommended a target have enough resources to handle a registration from each initiator the target has knowledge of.

In response to a PERSISTENT RESERVE OUT with a REGISTER service action the device server shall perform a registration by doing the following:

a) process the registration request regardless of any active persistent reservations;
b) map the reservation key to the registering initiator using the initiator identification and, if available, the initiator ports world wide identification;
c) register the reservation key without generating a persistent reservation; and
d) retain the reservation key and associated information.

After the registration request has been processed the device server shall then allow other PERSISTENT RESERVE OUT commands from the registered initiator to execute.

For each initiator that performs a PERSISTENT RESERVE OUT with a REGISTER service action, the device server shall retain the reservation key until the key is changed by a new PERSISTENT RESERVE OUT command with the REGISTER service action from the same initiator or until the initiator registration is removed (see clause 5.3.2.5).

Any PERSISTENT RESERVE OUT command service action received from an unregistered initiator, other than the REGISTER service action, results in the command being rejected with a RESERVATION CONFLICT status.

It is not an error for an initiator that is registered to reregister with the same reservation key or a new reservation key. A reregister shall have no effect any other registrations (i.e., when more than one initiator is registered with the same reservation key and one of those initiators reregisters it has no effect on the other initiators registrations). A reregister, not containing a Service action reservation key of zero, shall have no effect on any reservations (i.e., an initiator may change it's reservation key without affecting any previously created reservation).

Multiple initiators may register with the same reservation key. An initiator may use the same reservation key for multiple logical units. However, there shall only be one reservation key registered for each initiator/logical unit pair.

5.3.2.4 Creating a persistent reservation when there is no persistent reservation

An application client creates a persistent reservation by doing the following:

a) issuing a PERSISTENT RESERVE OUT command with a service action of RESERVE through a registered initiator; and
b) setting the reservation key of the registered initiator and the type and scope of the reservation being created.

Only one persistent reservation at a time per logical unit or element is allowed. If the target receives a PERSISTENT RESERVE OUT command that attempts to create a persistent reservation when a persistent reservation already exists for the logical unit from an initiator other than the initiator that created the reservation then the command shall be rejected with a RESERVATION CONFLICT status.

If the initiator that created the persistent reservation attempts to modify the type or scope of existing reservation then the command shall be rejected with a RESERVATION CONFLICT status.

If the device server receives a PERSISTENT RESERVE OUT command with a service action of RESERVE where the type and scope is the same as the existing type and scope from the initiator that created the persistent reservation it shall ignore the command and return a GOOD status.

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

See clause 5.3 for information on when a persistent reservation takes effect.

5.3.2.5 Removing registrations and persistent reservations

Any registered initiator may remove a persistent reservation by issuing one of the following commands:

a) a PERSISTENT RESERVE OUT command with a service action of RELEASE (see clause 5.3.2.5.1);
b) a PERSISTENT RESERVE OUT command with a PREEMPT service action (see clause 5.3.2.5.2.1);
c) a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action (see clause 5.3.2.5.2.2); or

d) a PERSISTENT RESERVE OUT command with a service action of CLEAR service action (see clause 5.3.2.5.3).

Any registered initiator may remove a registration by issuing one of the following commands:

a) a PERSISTENT RESERVE OUT command with a PREEMPT service action (see clause 5.3.2.5.2.1.3);
b) a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action (see clause 5.3.2.5.2.2); or
c) a PERSISTENT RESERVE OUT command with a CLEAR service action (see clause 5.3.2.5.3).
d) a PERSISTENT RESERVE OUT command with a REGISTER service action from the same initiator with the value of the SERVICE ACTION RESERVATION KEY field set to zero.

When a reservation key has been removed, no information shall be reported for that unregistered initiator in subsequent READ KEYS service action(s). Any persistent reservation associated with that unregistered initiator shall be released and if the released persistent reservation is of the type write exclusive, registrants only or exclusive access, registrants only the device server shall establish a unit attention condition for all the other registered initiators. The sense key shall be set to UNIT ATTENTION and the additional sense data shall be set to RESERVATIONS RELEASED.

An active persistent reservation may also be released by a power off. When the most recent APTPL value received by the device server is zero (see 7.12.2), a power off:

a) performs a hard reset;
b) clears all persistent reservations; and
c) removes all registered reservation keys (see 7.12.1.1).

5.3.2.5.1 Releasing a persistent reservation

Only the initiator that creates the persistent reservation is allowed to release that persistent reservation regardless of the type of persistent reservation.

An application client removes an active persistent reservation by doing the following:

a) Issuing a PERSISTENT RESERVE OUT command with a service action of RELEASE through the registered initiator that holds the persistent reservation; and
b) Setting the reservation key of the registered initiator and the type and scope to match the persistent reservation being released.

In response to a persistent reservation release request from an initiator that created the persistent reservation the device server shall perform a release by doing the following:

a) Removing the active persistent reservation; and
b) not removing any registration key(s).
c) If the released persistent reservation is of the type write exclusive, registrants only or exclusive access, registrants only the device server establishes a unit attention condition for all the other registered initiators. The sense key shall be set to UNIT ATTENTION and the additional sense data shall be set to RESERVATION RELEASED.

Editors Note 1 - GOP: RESERVATION RELEASED is a new ASCQ. This falls under the other note about when reservation go away.

---

d) If the persistent reservation is of any other type the device server establishes no unit attention condition.

The device server shall return a CHECK CONDITION status for a PERSISTENT RESERVE OUT command that specifies the release of a persistent reservation held by the requesting initiator if the scope and type does not match the scope and type of the persistent reservation being released. The sense key shall be set to ILLEGAL REQUEST and additional sense data shall be set to INVALID RELEASE OF ACTIVE PERSISTENT RESERVATION.

In response to a persistent reservation release request from an initiator that does not hold the persistent reservation or if there is no active persistent reservation the device server shall do the following:

a) Not remove the persistent reservation (if any);
b) not remove or change any registration key(s); and
c) return a status of GOOD.

~~A RELEASE service action should not be performed if any operations interlocked by the persistent reservation are not yet complete.~~

Editors Note 2 - GOP: The above paragraph is duplicate information form section 5.3.

5.3.2.5.2 Preempting an existing persistent reservation

5.3.2.5.2.1 PREEMPT service action

A PERSISTENT RESERVE OUT command with a PREEMPT service action is used to preempt a persistent reservation and/or registration. See figure 1 for a description of how a device server should interpret a PREEMPT service action to determine the actions it should take (i.e., preempt persistent reservation, preempt registration or preempt both registration and persistent reservation).

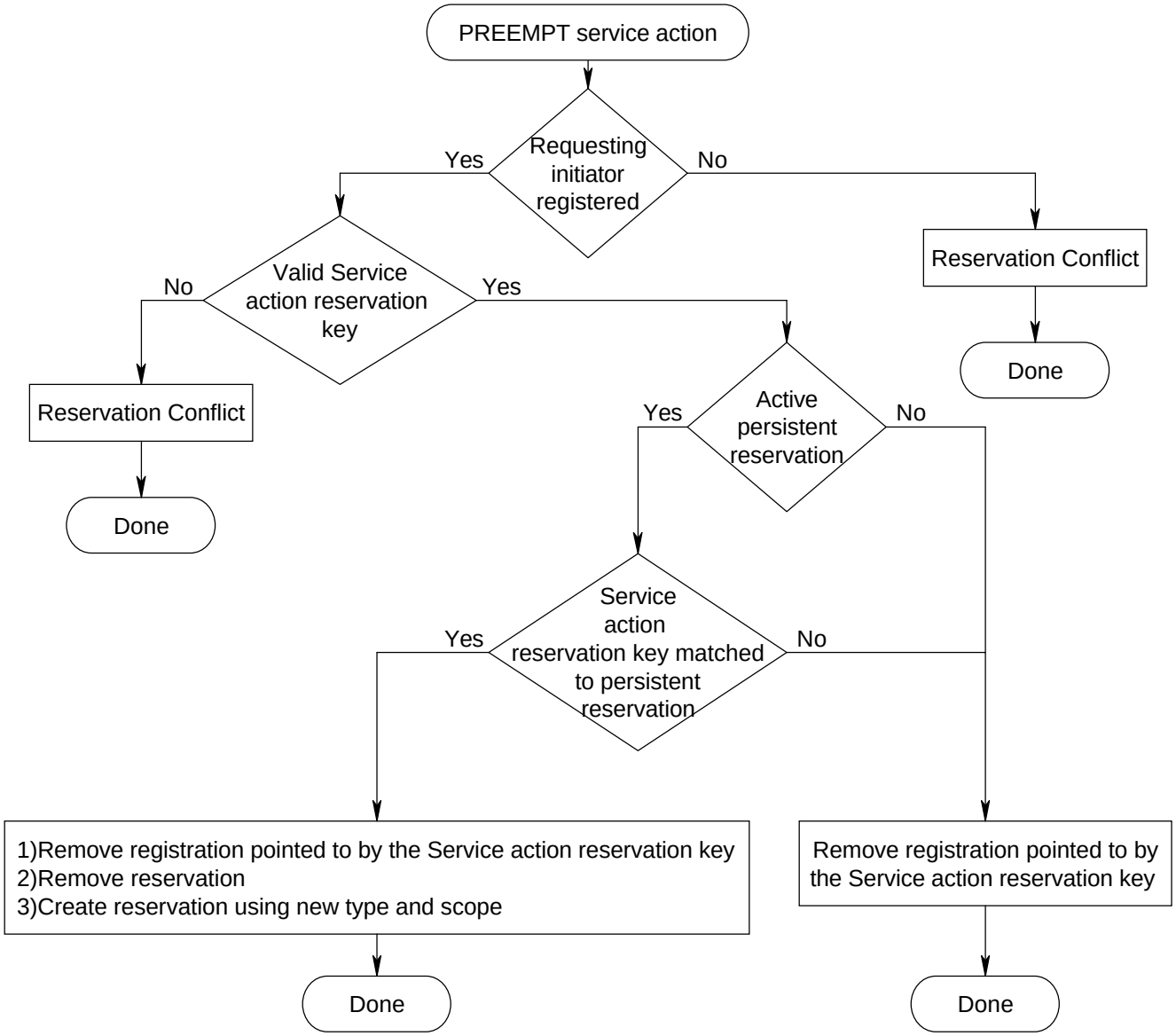

Figure 1 — Device server interpretation of PREEMPT service action

5.3.2.5.2.1.1 Failed persistent reservation preempt

If the preempting initiator's PREEMPT service action or PREEMPT AND ABORT service action fails it is recommended the preempting initiator issue a Logical Unit Reset task management function to the failing logical unit.

5.3.2.5.2.1.2 Preempting reservations

Any registered initiator may preempt any persistent reservation with another persistent reservation by doing the following:

a) issuing a PERSISTENT RESERVE OUT command with a PREEMPT service action through a registered initiator;

b) setting the Service action reservation key to match the reservation key of the persistent reservation being preempted; and
c) setting the type and scope of the new persistent reservation. The scope and type of the persistent reservation created by preempting initiator may be different than the persistent reservation being preempted.

If the Service action reservation key is associated with a reservation the device server shall perform a preempt by doing the following as an uninterrupted series of actions:

a) remove the persistent reservation for the initiator identified by the Service action reservation key specified in the PERSISTENT RESERVE OUT parameter list;
b) remove all registration(s) with a registration key that matches the Service action reservation key specified in the PERSISTENT RESERVE OUT parameter list (see 7.12.1.1);
c) establish a persistent reservation for the preempting initiator;
d) process tasks as defined in clause 5.3; and
e) establish a unit attention condition for any initiator that lost it's reservation and/or registration. The sense key shall be set to UNIT ATTENTION and the additional sense data shall be set to REGISTRATIONS PREEMPTED.

Subsequent tasks are subject to the persistent reservation restrictions established by the preempting initiator.

5.3.2.5.2.1.3 Preempting registrations

An application client may clear registrations without effecting a persistent reservation by doing the following:

a) issuing a PERSISTENT RESERVE OUT command with a PREEMPT service action through a registered initiator; and
b) setting the Service action reservation key to match the reservation key of the registration being cleared.

If the Service action reservation key is not associated with a reservation the device server shall perform a preempt by doing the following:

a) remove the registration for the initiator(s) identified by the Service action reservation key specified in the PERSISTENT RESERVE OUT parameter list;
b) process tasks as defined in clause 5.3; and
c) establish a unit attention condition for any initiator that lost it's registration. The sense key shall be set to UNIT ATTENTION and the additional sense data shall be set to REGISTRATIONS PREEMPTED.

Editors Note 3 - GOP: REGISTRATIONS PREEMPTED is a new ASC/ASCQ.

If a PERSISTENT RESERVE OUT specifying a PREEMPT service action sets the Service action reservation key to a value that does not match any registered reservation key the device server shall return a RESERVATION CONFLICT status.

A PERSISTENT RESERVE OUT specifying a PREEMPT service action with the Service action reservation key with a value equal to the reservation key is not an error.

5.3.2.5.2.2 PREEMPT AND ABORT service action

The initiators requests for and device server responses to a PERSISTENT RESERVE OUT command PREEMPT AND ABORT service action are identical to the PREEMPT service action (clause 5.3.2.5.2.1). In addition the device server shall do the following.

Every task from the preempted initiator shall be terminated as if an ABORT TASK task management function had been performed by the preempted initiator. If the key is registered and no persistent reservation exists for the initiator identified by the Service action reservation key the abort portion of the action shall terminate every commend from the preempted initiator.

Subsequent new tasks and retries of tasks that timed out because they were aborted are subject to the persistent reservation restrictions established by the preempting initiator.

The device server shall clear any ACA or CA condition associated with the initiator being preempted and shall clear any tasks with an ACA attribute from that initiator. ~~and return a status of GOOD~~. If TST=000b (see 8.3.4), then ACA or CA conditions for initiators other than the initiator being preempted shall prevent the execution of the PERSISTENT RESERVE OUT task. which shall end with a status of ACA ACTIVE if NACA=1 (see SAM), or BUSY if NACA=0. If TST=001b, then ACA or CA conditions for initiators other than the initiator being preempted shall not prevent the execution of the PERSISTENT RESERVE OUT task.

Any Asynchronous Event Reporting operations in progress that were initiated by the device server are not affected by the Preempt and Abort service action.

5.3.2.5.3 Clearing a persistent reservation

Any application client may clear a persistent reservation and all registrations from a device server for a specific logical unit by doing the following:

a) Issuing a PERSISTENT RESRVE OUT command with a service action of CLEAR service action through a registered initiator;

In response to this request the device server shall perform a clear by doing the following:

a) Remove the any persistent reservation associated with the logical unit;
b) remove all registration key(s) (see 7.12.1.1);
c) continue normal execution of any tasks from any initiator that have been accepted by the device server as nonconflicting; and
d) establish a Unit Attention condition for all initiators for the cleared logical unit. The sense key shall be set to UNIT ATTENTION and the additional sense data shall be set to RESERVATION CLEARED.

Editors Note 4 - GOP: RESERVATION CLEARED is a new ASCQ.

Application clients should not use the Clear service action except during recoveries that are associated with initiator or system reconfiguration, since data integrity may be compromised.

7.11 PERSISTENT RESERVE IN command

The PERSISTENT RESERVE IN command (see table 1) is used to obtain information about persistent reservations and reservation keys that are active within a device server. This command is used in conjunction with the PERSISTENT RESERVE OUT command (see 7.12).

Table 1 — PRESISTENT RESERVE IN command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (5Eh)							
1	RESERVED			SERVICE ACTION				
2	RESERVED							
3	RESERVED							
4	RESERVED							
5	RESERVED							
6	RESERVED							
7	(MSB)	ALLOCATION LENGTH						
8								(LSB)
9	CONTROL							

When a device server receives a PERSISTENT RESERVE IN command and RESERVE(10) or RESERVE(6) logical unit or SMC element reservations are active (see 7.21), the command shall be rejected with a RESERVATION CONFLICT status. A PERSISTENT RESERVE IN command shall not conflict with any persistent reservation even if the initiator is not registered.

The actual length of the PERSISTENT RESERVE IN parameter data is available in a parameter data field. The ALLOCATION LENGTH field in the CDB indicates how much space has been reserved for the returned parameter list. If the length is not sufficient to contain the entire parameter list, the first portion of the list shall be returned. This shall not be considered an error. If the remainder of the list is required, the application client should send a new PERSISTENT RESERVE IN command with a ALLOCATION LENGTH field large enough to contain the entire list.

7.11.1 PERSISTENT RESERVE IN Service Actions

The SERVICE ACTION codes for the PERSISTENT RESERVE IN command are defined in table 2.

Table 2 — PERSISTENT RESERVE IN SERVICE ACTION codes

Code	Name	Description
00h	READ KEYS	Reads all registered Reservation Keys
01h	READ RESERVATION	Reads the current persistent reservation
02h - 1Fh	Reserved	Reserved

7.11.1.1 Read Keys

The READ KEYS service action requests that the device server return a parameter list containing a header and a complete list of all reservation keys currently registered with the device server. If multiple initiators have registered with the same key, then that key value shall be listed multiple times, once for each such registration. The keys may

have been passed by a PERSISTENT RESERVE OUT command that has performed a Register service action. READ RESERVATION.

For more information on read keys see clause 5.3.2.2.1.

7.11.1.2 Read Reservation

The Read Reservation service action requests that the device server return a parameter list containing a header and the persistent reservation that is presently active in the device server.

For more information on read reservation see clause 5.3.2.2.2.

7.11.2 PERSISTENT RESERVE IN parameter data for Read Keys

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the Read Keys service action is shown in table 3.

Table 3 — PERSISTENT RESERVE IN parameter data for Read Keys

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3				GENERATION				(LSB)
4	(MSB)							
7				ADDITIONAL LENGTH (n-7)				(LSB)
				Reservation key list				
8	(MSB)							
15				RESERVATION KEY (first)				(LSB)
				.				
				.				
				.				
n-7	(MSB)							
n				RESERVATION KEY (last)				(LSB)

The GENERATION field contains a 32-bit counter in the device server that shall be incremental every time a PERSISTENT RESERVE OUT command requests a Register, a Clear, a Preempt, or a Preempt and Abort service action. The counter shall not be incremental by a PERSISTENT RESERVE IN command, by a PERSISTENT RESERVE OUT command that performs a Reserve or Release service action, or by a PERSISTENT RESERVE OUT command that is not performed due to an error or reservation conflict. Regardless of the APTPL value the generation value shall be set to 0 as part of the power on reset process.

The ADDITIONAL LENGTH field contains a count of the number of bytes in the Reservation key list. If the allocation length specified by the PERSISTENT RESERVE IN command is not sufficient to contain the entire parameter list, then only the bytes from 0 to the maximum allowed allocation length shall be sent to the application client. The incremental remaining bytes shall be truncated, although the ADDITIONAL LENGTH field shall still contain the actual number of bytes in the reservation key list without consideration of any truncation resulting from an insufficient allocation length. This shall not be considered an error.

The Reservation key list contains all the 8-byte reservation keys registered with the device server through a Register service action.

7.11.3 PERSISTENT RESERVE IN parameter data for Read Reservation

The format for the parameter data provided in response to a PERSISTENT RESERVE IN command with the Read Reservation service action is shown in table 4.

Table 4 — PERSISTENT RESERVE IN parameter data for Read Reservations

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
3				GENERATION				(LSB)
4	(MSB)							
7				ADDITIONAL LENGTH (n-7)				(LSB)
8	(MSB)							
n				RESERVATION DESCRIPTORS				(LSB)

The GENERATION field shall be as defined for the PERSISTENT RESERVE IN Read Keys parameter data (see 7.11.2).

The ADDITIONAL LENGTH field contains a count of the number of bytes to follow in the RESERVATION DESCRIPTOR(s). If the allocation length specified by the PERSISTENT RESERVE IN command is not sufficient to contain the entire parameter list, then only the bytes from 0 to the maximum allowed allocation length shall be sent to the application client. The remaining bytes shall be truncated, although the ADDITIONAL LENGTH field shall still contain the actual number of bytes of the RESERVATION DESCRIPTOR(s) and shall not be affected by the truncation. This shall not be considered an error.

The format of a single read Reservation descriptor is defined in table 5. There shall be one read Reservation descriptor for the persistent reservation held on the logical unit or, in the case of multiple elements, one read Reservation descriptor for each element that holds a persistent reservation.

Table 5 — PERSISTENT RESERVE IN Read Reservations Descriptor

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)							
7				RESERVATION KEY				(LSB)
8	(MSB)							
11				SCOPE-SPECIFIC ADDRESS				(LSB)
12	RESERVED							
13	SCOPE				TYPE			
14	RESERVED							
15	RESERVED							

If a persistent reservation is held on the logical unit that does not contain elements, there shall be a single read reservation descriptor presented in the list of parameter data returned by the device server in response to the PERSISTENT RESERVE IN command with a READ RESERVATION service action. The read reservation descriptor shall contain the Reservation Key under which the persistent reservation is held. The type and scope of the persistent reservation as present in the PERSISTENT RESERVE OUT command that created the persistent reservation shall be returned (see 7.11.3.1 and 7.11.3.2).

If a persistent reservation is held on the logical unit that does contain elements, there shall be a read reservation descriptor presented in the list of parameter data returned by the device server in response to the PERSISTENT RESERVE IN command with a READ RESERVATION service action for each persistent reservation. The read reservation descriptor shall contain the Reservation Key under which the persistent reservation is held. The type and scope of the persistent reservation as present in the PERSISTENT RESERVE OUT command that created the persistent reservation shall be returned (see 7.11.3.1 and 7.11.3.2).

If the scope is an Element reservation, the scope-specific address field shall contain the Element address, zero filled in the most significant bytes to fit the field. If the scope is a Logical Unit reservation the scope-specific address shall be set to zero.

7.11.3.1 Persistent Reservations Scope

The value in the scope field shall indicate whether a persistent reservation applies to an entire logical unit or to an element. The values in the scope field are defined in table 6.

Table 6 — Persistent Reservation Scope Codes

Code	Name	Description
0h	LU	Persistent reservation applies to the full logical unit
1h	Reserved	Obsolete
2h	Element	Persistent reservation applies to the specified element
3h - Fh	Reserved	Reserved

7.11.3.1.1 LU Scope

A SCOPE field value of LU shall indicate that the persistent reservation applies to the entire logical unit. The LU scope shall be implemented by all device servers that implement PERSISTENT RESERVE OUT.

7.11.3.1.2 Element Scope

A SCOPE field value of Element shall indicate that the persistent reservation applies to the element of the logical unit defined by the scope-specific address field in the PERSISTENT RESERVE OUT parameter list. An element is defined by the SCSI-3 Medium Changer Commands (SMC) standard. The Element scope is optional for all device servers that implement PERSISTENT RESERVE OUT.

7.11.3.2 Persistent Reservations Type

The value in the TYPE field shall specify the characteristics of the persistent reservation being established for all data blocks within the element or within the logical unit. Table 7 defines the characteristics of the different type values. For each persistent reservation type, table 7 lists code value and describes the required device server support. In table 7, the description of required device server support is divided into two paragraphs. The first

paragraph defines the required handling for read operations. The second paragraph defines the required handling for write operations.

Table 7 — Persistent Reservation Type Codes

Code	Name	Description
0h	Reserved	Obsolete
1h	Write Exclusive	**Reads Shared**: Any application client on any initiator may execute tasks that perform transfers from the storage medium or cache of the logical unit to the initiator. **Writes Exclusive**: Any task from any initiator other than the initiator holding the persistent reservation that performs a transfer from the initiator to the storage medium or cache of the logical unit shall result in a reservation conflict.
2h	Reserved	Obsolete
3h	Exclusive Access	**Reads Exclusive**: Any task from any initiator other than the initiator holding the persistent reservation that performs a transfer from the storage medium or cache of the logical unit to the initiator shall result in a reservation conflict. **Writes Exclusive**: Any task from any initiator other than the initiator holding the persistent reservation that performs a transfer from the initiator to the storage medium or cache of the logical unit shall result in a reservation conflict.
4h	Reserved	Obsolete
5h	Write Exclusive, Registrants Only	**Reads Shared**: Any application client on any initiator may execute tasks that perform transfers from the storage medium or cache of the logical unit to the initiator. **Writes Exclusive**: Any task from an initiator that has not previously performed a Register service action with the device server that performs a transfer to the storage medium or cache of the logical unit, shall result in a reservation conflict.
6h	Exclusive Access, Registrants Only	**Reads Exclusive**: Any task from an initiator that has not previously performed a Register service action with the device server that performs a transfer from the storage medium or cache of the logical unit, shall result in a reservation conflict. **Writes Exclusive**: Any task from an initiator that has not previously performed a Register service action with the device server that performs a transfer to the storage medium or cache of the logical unit, shall result in a reservation conflict.
7h - Fh	Reserved	

7.12 PERSISTENT RESERVE OUT command

The PERSISTENT RESERVE OUT command (see table 8) is used to reserve a logical unit or element for the exclusive or shared use of a particular initiator. The command shall be used in conjunction with the PERSISTENT RESERVE IN command and shall not be used with the RESERVE and RELEASE commands.

Persistent reservations shall conflict with reservations established by the RESERVE command. Initiators performing PERSISTENT RESERVE OUT service actions are identified by a reservation key provided by the application client. An application client may use the PERSISTENT RESERVE IN command to identify which initiators is holding a persistent reservations and use the PERSISTENT RESERVE OUT command to preempt that reservations if required.

Table 8 — PRESISTENT RESERVE OUT command

Bit / Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE (5Fh)							
1	RESERVED				SERVICE ACTION			
2	SCOPE				TYPE			
3	RESERVED							
4	RESERVED							
5	RESERVED							
6	RESERVED							
7	(MSB)	PARAMETER LIST LENGTH (18h)						
8								(LSB)
9	CONTROL							

When a device server receives a PERSISTENT RESERVE OUT command and RESERVE(10) or RESERVE(6) logical unit or SMC element reservations are active (see 7.21), the command shall be rejected with a RESERVATION CONFLICT status.

If a PERSISTENT RESERVE OUT command is attempted, but there are insufficient device server resources to complete the operation, the device server shall return a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense data shall be set to INSUFFICIENT REGISTRATION RESOURCES.

Editors Note 5 - GOP: Change the INSUFFICIENT RESERVATION RESOURCES ASCQ to INSUFFICIENT REGISTRATION RESOURCES.

The PERSISTENT RESERVE OUT command contains fields that specify a persistent reservation service action, the intended scope of the persistent reservation, and the restrictions caused by the persistent reservation. The TYPE and SCOPE fields are defined in 7.11.3.1 and 7.11.3.2. If a SCOPE field specifies a scope that is not implemented, the device server shall return a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and additional sense data shall be set to INVALID FIELD IN CDB.

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

The parameter list shall be 24 bytes in length and the Parameter list length field shall contain 24 (18h). If the Parameter list length is not 24, the device server shall return a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense data shall be set to PARAMETER LIST LENGTH ERROR.

7.12.1 PERSISTENT RESERVE OUT Service Actions

When processing the PERSISTENT RESERVE OUT service actions, the device server shall increment the generation value as specified in 7.11.2.

The PERSISTENT RESERVE OUT command service actions are defined in table 9.

Table 9 — PERSISTENT RESERVE OUT SERVICE ACTION codes

Code	Name	Description
00h	REGISTER	Register a reservation key with the device server
01h	RESERVE	Create a persistent reservation using a reservation key
02h	RELEASE	Release a persistent reservation
03h	CLEAR	Clear all reservation keys and all persistent reservations
04h	PREEMPT	Preempt persistent reservations from another initiator
05h	PREEMPT & ABORT	Preempt persistent reservations from another initiator and abort the task set for the preempted initiator
06h - 1Fh	Reserved	

7.12.1.1 Register

When a reservation key has not yet been established or when the reservation key has been removed, the RESERVATION KEY field shall be set to zero when the initiator performs a PERSISTENT RESERVE OUT with the REGISTER service action.

Any persistent reservations for the initiator shall be updated to reflect the new reservation key. The device server shall ignore the contents of the SCOPE and TYPE fields.

For more information on registering see clause 5.3.2.3.

7.12.1.2 Reserve

The PERSISTENT RESERVE OUT command performing a Reserve service action creates a persistent reservation having a specified scope and type. The scope and type of a persistent reservation are defined in 7.11.3.1 and 7.11.3.2.

For more information on reserving see clause 5.3.2.4.

7.12.1.3 Release

The PERSISTENT RESERVE OUT command performing a Release service action removes the selected reservation for the requesting initiator.

For more information on releasing see clause 5.3.2.5.1.

7.12.1.4 Clear

The device server shall ignore the contents of the SCOPE and TYPE fields.

For more information on clearing see clause 5.3.2.5.3.

7.12.1.5 Preempt

The persistent reservation created by the preempting initiator is specified by the scope and type field of the PERSISTENT RESERVE OUT command and the corresponding fields in the PERSISTENT RESERVE OUT parameter list.

For more information on preempting see clause 5.3.2.5.2.1.

7.12.1.6 Preempt and Abort

The PERSISTENT RESERVE OUT command performing a Preempt and Abort service action have the same requirements as defined for the Preempt service action (see clause 7.12.1.6).

For more information on preempting and aborting see clause 5.3.2.5.2.1 and clause 5.3.2.5.2.2.

7.12.2 PERSISTENT RESERVE OUT parameter list

The parameter list required to perform the PERSISTENT RESERVE OUT command are defined in table 10. All fields shall be sent on all PERSISTENT RESERVE OUT commands, even if the field is not required for the specified SERVICE ACTION and scope values.

Table 10 — PERSISTENT RESERVE OUT parameter list

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB)			RESERVATION KEY				
7								(LSB)
8	(MSB)			SERVICE ACTION RESERVATION KEY				
15								(LSB)
16	(MSB)			SCOPE-SPECIFIC ADDRESS				
19								(LSB)
20	RESERVED							APTPL
21	RESERVED							
21	RESERVED							
21	RESERVED							

The RESERVATION KEY field contains an 8-byte value provided by the application client to the device server to identify the initiator that is the source of the PERSISTENT RESERVE OUT command. The device server shall verify that the RESERVATION KEY field in a PERSISTENT RESERVE OUT command matches the registered reservation key for the initiator from which the task was received except for the Register service action for an unregistered initiator which shall have a reservation key value of zero. If a PERSISTENT RESERVE OUT command specifies a RESERVATION KEY field other than the reservation key registered for the initiator, the device server shall

return a RESERVATION CONFLICT status. The reservation key of the initiator shall be valid for all service action and scope values.

The SERVICE ACTION RESERVATION KEY field contains information needed for the following service actions; the Register, Preempt, and Preempt and Abort service actions. For the Register service action, the SERVICE ACTION RESERVATION KEY field contains the new reservation key to be registered. For the Preempt and Preempt and Abort service actions, the SERVICE ACTION RESERVATION KEY field contains the reservation key of the persistent reservations that are being preempted. The service action reservation key is ignored for all service actions except those described in this paragraph.

If the scope is an Element reservation, the scope-specific address field shall contain the Element address, zero filled in the most significant bytes to fit the field. If the service action is Register or Clear or if the scope is a logical unit reservation, the SCOPE-SPECIFIC ADDRESS field shall be set to zero.

The Activate Persist Through Power Loss (APTPL) bit shall be valid only for the Register service action. In all other cases, the APTPL bit shall be ignored. Support for an APTPL bit equal to one is optional. If a device server that does not support the APTPL bit value of one receives that value in a Register service action, the device server shall return a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and additional sense data shall be set to INVALID FIELD IN PARAMETER LIST.

If the last valid APTPL bit value received by the device server is zero, the loss of power in the target shall release the persistent reservation for all logical units and remove all reservation keys (see 7.12.1.1). If the last valid APTPL bit value received by the device server is one, the logical unit shall retain the persistent reservation for all logical units and all reservation keys for all initiators even if power is lost and later returned.

Table 46 summarizes which fields are set by the application client and interpreted by the device server for each service action and scope value. Two PERSISTENT RESERVE OUT parameters are not summarized in table 46; reservation key and APTPL, since they are specified above.

Table 11 — PERSISTENT RESERVE OUT SERVICE ACTIONS and valid parameters

Service Action	Allowed Scope	Parameters		
		Type	Service Action Reservation key	Element or Element Parameters
Register	ignored	ignored	valid	ignored
Reserve Reserve	LU Element	valid valid	ignored ignored	ignored Element valid
Release Release	LU Element	valid valid	ignored ignored	ignored Element valid
Clear	ignored	ignored	ignored	ignored
Preempt Preempt	LU Element	valid valid	valid valid	ignored Element valid
Preempt & abort Preempt & abort	LU Element	valid valid	valid valid	ignored Element valid

7.12.3 Another ASCQ

There are times, especially with dual port, when an active task has ownership of drive resources but is not responding or is responding very slowly. We would like to be able to abort the task we are working on for this initiator and start another one for the other port or another initiator on the same port.

We would like a RESOURCE CONFLICT ASCQ. We would use it with the ABORTED COMMAND key and the recovery action would be to retry the command.

I suggest this new ASCQ be placed under the SYSTEM RESOURCES FAILURE (55h 00h) ASC.