T10/98-203 revision 4

Date: Sept 11, 1998

To: T10 Committee (SCSI)

From: George Penokie (IBM)

Subject: Persistent Reserations

0.0.1 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. 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 are 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.

Editors Note 1 - GOP: Should a persistent reservation and/or a registration be held if an initiator explicitly logs out? One answer is no because that initiator is taking an explicate action and should have removed the persistent reservation and/or registration before logging out. (i.e, it is an error to log out if you have an outstanding persistent reservation or registration). Another answer is to keep it across log out in all cases because that is no different than a reset.

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 makes it possible to determine which port holds the ~~conflicting~~ persistent reservation and to take over a persistent reservation from a failing or uncooperative initiator.

~~Persistent reservation service actions that require access to the persistent reservation and registration information may require accessing a nonvolatile memory within the logical unit. Any SCSI device that supports persistent reservation and has If the nonvolatile memory that is not ready, the device server shall return CHECK CONDITION status for all xxx type commands. 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).~~

Editors Note 2 - GOP: Above move to end of section 0.0.1.1

0.0.1.1 Preserving persistent reservations across power cycles

The application client many 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 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 identification;
b) reservation key; and
c) if available, initiator ports world wide identification.

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

d) Initiator identification;
e) reservation key;
a) scope;
b) type; and
c) if available, initiator ports world wide identification.

When available, the initiator ports world wide identification shall be used to determine if the initiator identification has changed. If the initiator identification changed the device server shall assign the new initiator identification to the existing registration and reservation.

The capability of preserving persistent reservations and registration keys across power cycles requires the use of a nonvolatile memory within the SCSI device. ~~Persistent reservation service actions that require access to the persistent reservation and registration information may require accessing a nonvolatile memory within the logical unit.~~ Any SCSI device that supports the Persist Through Power Loss (APRPL) capability of persistent reservation and has ~~If the~~ nonvolatile memory that is not ready~~, the device server~~ shall return CHECK CONDITION status for all xxx type commands. 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).

Editors Note 3 - GOP: The above changes make a stronger statement about not accepting any tasks before the persistent reservation information is accessible (i.e.,placed into nonvolatile memory). There is a question about what tasks should be allowed, if any.

0.0.1.2 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.

0.0.1.2.1 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 reservation read keys request from an initiator the device server shall report the following if there are any registered initiators:

a) the current generation counter value; and
b) ~~all the registered~~ 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. ~~, however, multiple keys are not registered to a single initiator.~~

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 4 - GOP: How about something that states a target should have enough resources to handle x number of registrations (e.g. one per initiator in the same way we require the target to accept at least one task per initiator).

0.0.1.2.2 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 determine if a~~ To receive the persistent reservation ~~is active and~~ information ~~about any persistent reservations on a device server the~~ application client shall:

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

In response to a persistent reservation read reservation request from an initiator the device server shall report the following if a persistent reservation is active:

a) the current generation counter value;
b) the registered reservation key associated with the initiator that holds the active persistent reservation; and
c) the scope and type of each active persistent reservation.

An application client may use the reservation key to associate the persistent reservation with the initiator that holds the persistent reservation. ~~Since initiators use unique reservation keys, the application client should be able to associate the reservation key with the initiator that holds the reservation.~~ This association is done using techniques that are outside the scope of this standard.

0.0.1.3 Registering

To establish a persistent reservation the initiator must 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 reservation key. ~~which the device server retains and maps to the registering initiator.~~

In response to a persistent reservation register request from an initiator 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.

~~The device server shall process the registration request regardless of any active persistent reservations and shall register the reservation key without generating any persistent reservations.~~ 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 by one of the following actions:

a) powering down the logical unit, if the last APTPL received by the device server was zero (see 0.2.2);
b) performing a CLEAR service action;
c) performing a PREEMPT service action;
d) performing a PREEMPT AND ABORT service action; or
e) performing 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 the READ KEYS service action and any persistent reservation associated with that initiator and reservation key pair shall be released.

Editors Note 5 - GOP: In the case where an initiator that holds the reservation unregisters the reservation shall be released!?! And if this is the case then 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 RESERVATIONS RELEASED. The behavior is that when a reservation is released then all the other initiators tasks start or keep execution because there is now no present reservation.

Any PERSISTENT RESERVE OUT command service action received by 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).

It is not an error for the same reservation key to be registered to multiple initiators or for an initiator to use the same reservation key for multiple logical units. ~~, however, multiple keys shall not be registered to a single initiator.~~

0.0.1.4 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 were 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.

Editors Note 6 - GOP: The above paragraph has been modified to allow an initiator to issue a persistent reservation when a persistent reservation already exists when that initiator holds the persistent reservation. This is the same behavior that normal reservation has.

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.

A persistent reservation shall take effect when the task executing the PERSISTENT RESERVE OUT command enters the enabled task state.

0.0.1.5 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 0.0.1.5.1);
b) a PERSISTENT RESERVE OUT command with a PREEMPT service action (see clause 0.0.1.5.2.1);
c) a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action (see clause 0.0.1.5.2.2); or
d) a PERSISTENT RESERVE OUT command with a service action of CLEAR service action (see clause 0.0.1.5.3).

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 0.0.1.5.2.1.3);
b) a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action (see clause 0.0.1.5.2.2); or
c) a PERSISTENT RESERVE OUT command with a service action of CLEAR service action (see clause 0.0.1.5.3).

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 0.2.2), a power off:

a) performs a hard reset;
b) clears all persistent reservations; and
c) removes all registered reservation keys (see 0.2.1.1).

0.0.1.5.1 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 7 - 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. ~~matching some but not all of the scope and type. reservation key, and extent values.~~ 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 doing 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.

~~An active persistent reservation may also be released by either of the following mechanisms:~~

~~a) Power off. When the most recent APTPL value received by the device server is zero (see 0.2.2), a power off performs a hard reset, clears all persistent reservations, and removes all registered reservation keys (see 0.2.1.1); or~~

~~b) Execution of a PERSISTENT RESERVE OUT command from another initiator with a Persistent Reserve service action of PREEMPT, PREEMPT AND ABORT, or CLEAR.~~

Editors Note 8 - GOP: The above was moved to section 0.0.1.5

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

0.0.1.5.2 Preempting an existing persistent reservation

0.0.1.5.2.1 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).

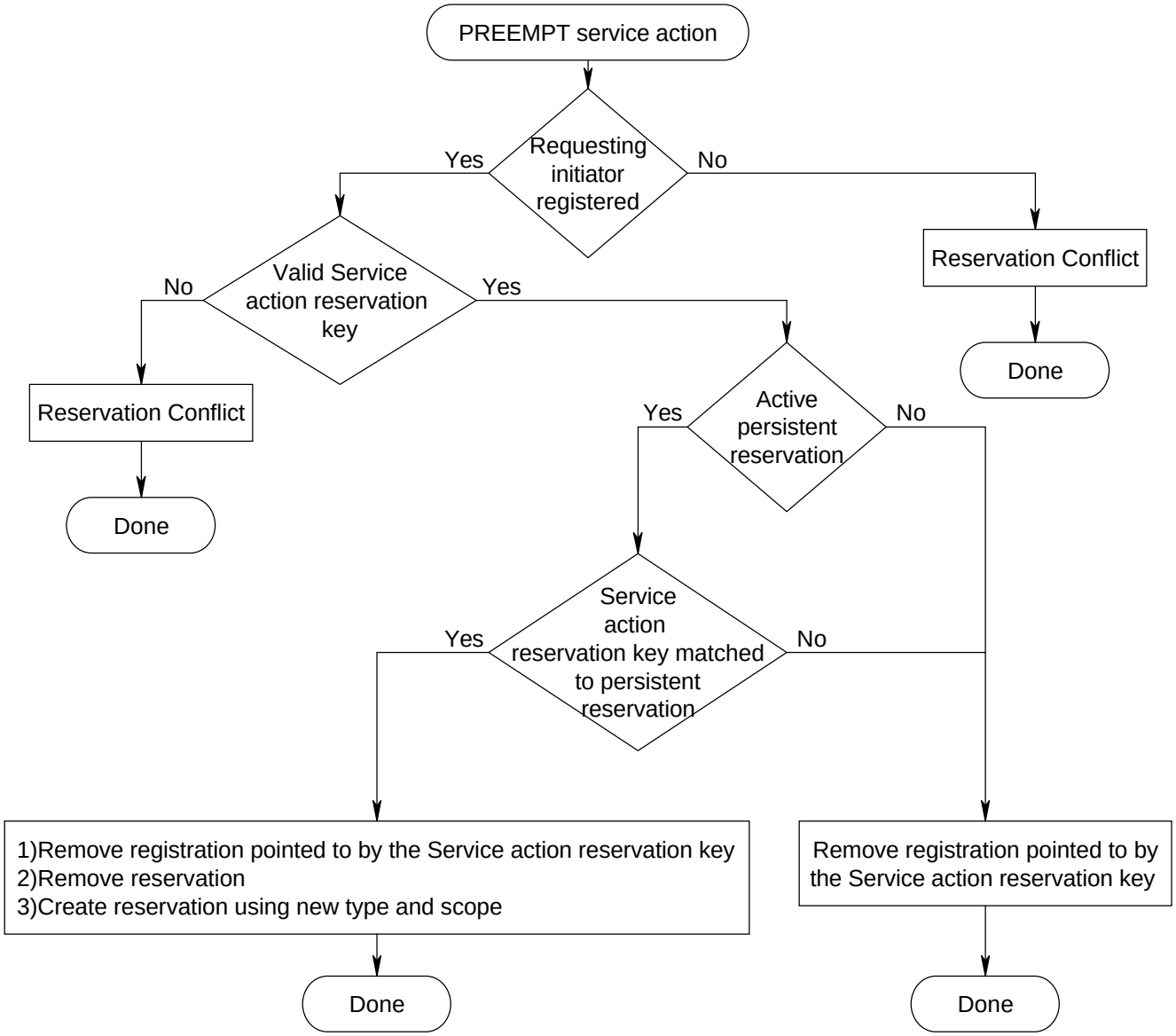

Figure 1 — Device server interpretation of PREEMPT service action

0.0.1.5.2.1.1 Termination of preempted initiators pending tasks

If the task manager has pending tasks for the preempted initiator the task manager shall, if possible, transition all those pending tasks into the ended state. As a result the preempted initiator shall receive a status of RESERVATION CONFLICT for each task that transitioned from the dormant state or blocked state to the ended state. Each task that transitioned from the enabled state to the ended state shall be terminated with a CHECK CONDITION status. The sense key shall be set to COMMAND ABORTED and the additional sense data shall be set to COMMANDS CLEARED BY ANOTHER INITIATOR.

If the preempting initiator's PREEMPT service action fails it is recommended the preempting initiator issue a PREEMPT AND ABORT service action with a HEAD OF QUEUE task attribute. The PREEMPT AND ABORT service action removes any tasks that are blocking the execution of preempting initiator's tasks.

0.0.1.5.2.1.2 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:

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 0.2.1.1);
c) establish a persistent reservation for the preempting initiator;
d) continue ~~normal~~ execution of ~~any~~ tasks as defined in clause 0.0.1.5.2.1.1; ~~from any initiator that were accepted by the device server as nonconflicting when the old persistent reservation was active~~;

Editors Note 9 - GOP: The fact that you allow old tasks to execute under the old reservation implies that the only sane way to do this is to treat the preempt as an ordered task. This means that if there is a stuck host, It may never get to this task. This is a deadlock condition. We can solve this by putting all of the tasks under the new reservation. Is this desirable or even acceptable? See clause 0.0.1.5.2.1.1 for a possible solution to this problem.

Editors Note 10 - GOP: What is supposed to happen to the tasks (if any) that were accepted from the initiator that is being preempted?

e) establish a unit attention condition for any initiator that lost it's reservation and registration. The sense key shall be set to UNIT ATTENTION and the additional sense data shall be set to RESERVATIONS PREEMPTED; and
f) establish a unit attention condition for any initiator that only looses it's 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.

0.0.1.5.2.1.3 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) continue execution of tasks as defined in clause 0.0.1.5.2.1.1; and

Editors Note 11 - GOP: The fact that you allow old tasks to execute under the old reservation implies that the only sane way to do this is to treat the preempt as an ordered task. This means that if there is a stuck host, It may never get to this task. This is a deadlock condition. We can solve this by putting all of the tasks under the new reservation. Is this desirable or even acceptable? See clause 0.0.1.5.2.1.1 for a possible solution to this problem.

Editors Note 12 - GOP: What is supposed to happen to the tasks (if any) that were accepted from the initiator that is being preempted?

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 13 - 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.

~~If a PERSISTENT RESERVE OUT specifying a PREEMPT service action sets the Service action reservation key to a value equal to the Reservation key the device server shall return a RESERVATION CONFLICT status.~~

Editors Note 14 - GOP: The above paragraph has been removed because it prevents an initiator from preempting itself. If we do not allow this behavior then we get into a case were there is a reservation but no initiator holds that reservation. The reason no one holds the reservation is because if the key of the preempted and preempting initiators is the same then the key of the preempted initiator is cleared. Therefore the preempting initiator now has no key because they were the same key.

0.0.1.5.2.2 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 0.0.1.5.2.1). 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 ~~execute normally~~ 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.

Editors Note 15 - GOP: In the above paragraph a status of GOOD is returned for an ACA task active when the about occurs. This does not seem correct. It should return a CHECK CONDITION with a key of ABORTED COMMAND.

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

0.0.1.5.3 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 0.2.1.1);
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 16 - 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.

0.1 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 0.2).

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.

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.

0.1.1 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

0.1.1.1 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 0.0.1.2.1.

0.1.1.2 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 0.0.1.2.2.

0.1.2 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 conains 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.

0.1.3 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 0.1.2).

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 contains a unique 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							

~~The Additional length field contains a count of the number of bytes to follow in the parameter 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 remaining bytes shall be truncated, although the Additional length field shall still contain the actual number of bytes and shall not be affected by the truncation. This shall not be considered an error.~~

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 0.1.3.1 and 0.1.3.2).

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 0.1.3.1 and 0.1.3.2).

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.

0.1.3.1 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

0.1.3.1.1 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.

0.1.3.1.2 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.

0.1.3.2 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	

0.2 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 17 - 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 0.1.3.1 and 0.1.3.2. 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.

0.2.1 PERSISTENT RESERVE OUT Service Actions

When processing the PERSISTENT RESERVE OUT service actions, the device server shall increment the generation value as specified in 0.1.2.

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	

0.2.1.1 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 0.0.1.3.

0.2.1.2 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 0.1.3.1 and 0.1.3.2.

For more information on reserving see clause 0.0.1.4.

0.2.1.3 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 0.0.1.5.1.

0.2.1.4 Clear

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

For more information on clearing see clause 0.0.1.5.3.

0.2.1.5 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 0.0.1.5.2.1.

0.2.1.6 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 0.2.1.6).

For more information on preempting and aborting see clause 0.0.1.5.2.1 and clause 0.0.1.5.2.2.

0.2.2 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 0.2.1.1). 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

0.2.3 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.