Sun Microsystems, Inc.
2550 Garcia Avenue
Mountain View, CA 94045
415 960-1300

April 18, 1998

% Sun
Logo
for

John Lohmeyer
Chairperson, X3T10
Symbios Logic Inc.
1635 Aeroplaza Drive
Colorado Springs, Colorado 80916

Subject: Proposal for Persistent Reservation

The following proposal has been extracted from work by Bill Dallas of DEC, Roger Cummings of Storage Technology, and many others. The proposal extends the definition of reservations to allow proper behavior in multi-initiator and multi-port environments. The proposal defines persistent reservations which remain valid across Target Reset and can only be cleared by power down or by a properly qualified persistent reservation from another initiator. Using the commands defined by this proposal, a host can protect the logical unit from improper behavior caused by another initiator on the same or other ports. At the same time, the host can determine from a logical unit which initiators share the logical unit, which initiator is presently reserving the logical unit, and can choose to displace the reservation of an initiator which is known to have failed.

Sincerely,

Robert N. Snively
Sun Microsystems
Mail Stop MPK 12-204
2550 Garcia Ave.
Mountain View, CA 94043-1100
phone: 415-786-6694
e-mail: bob.snively@sun.com

Persistent Reservation Proposal

Description of function / Persistent Reservation model:

Two new commands are defined in SPC, PERSISTENT RESERVE IN (PRIN) and PERSISTENT RESERVE OUT (PROUT). The commands are used to create and release persistent reservations (including logical unit or Extent reservations), to provide reservation keys to logical units, and to force actions on tasks from other initiators.

The PRIN and PROUT commands have the following capabilities, many of which are optional:

- Create persistent reservations using PROUT.
  - Extent (shared reservations defined)
  - Exclusive logical unit
  - Shared logical unit (shared reservations defined)
- Release persistent reservations using PROUT.
- Register an 8-byte Reservation Key for the sourcing initiator with the attached device using PROUT.
- Determine the present reservation's key and characteristics using PRIN.
- Determine the key of other initiators that are attached to the peripheral device using PRIN.
- Preempt a reservation with another initiator with new reservation from this using PROUT.
- Optionally automatically clear all tasks related with the preempted reservation.
- Generate a 32-bit number increasing with each reservation to warn of intervening actions.

A persistent reservation is formed between an initiator and a logical unit when the PROUT command containing the parameters requesting a persistent reservation is successfully executed. The parameters indicate the type of reservation that is formed. In addition, the parameters contain an 8-byte reservation key that is used by the target and by software on other initiators to identify the initiator holding the reservation. The persistent reservation cannot be released by a Target Reset, other reset activity, or by a RELEASE command. It can be released only by a PROUT from the same initiator containing the release parameters, by a power off, or by a PROUT from another initiator containing the proper preemptive reservation parameters and the reservation key of the initiator holding the reservation. While the persistent reservation is active, any conflicting persistent reservation or activity that conflicts with the reservation is rejected with RESERVATION CONFLICT status. This behavior allows any cooperating initiators having access to the logical unit through any port to execute carefully managed reservation protocols that will allow the logical unit to be safely shared among them. The reservation is safe from interruption by any initiators not participating in the persistent reservation protocol, even during booting and error recovery operations.

The use of a reservation key as part of the reservation process enables initiators to identify other ports holding reservations or sharing the logical unit. Hosts use the reservation key to perform locking and failover recovery operations. The communication for such algorithms is usually through an auxiliary port such as Ethernet or Fibre Channel. Targets with reservations held by failing hosts can be identified and preemptively reserved by hosts that are still operational. When preemptive reservations are performed, the reserving initiator can optionally invoke an automatic clearing of all tasks for the initiator port that is being preempted.

The structure of PROUT favors the generation of a single reservation for each PROUT command. Alternative implementations are discussed at the end of this proposal.

The reservation key of other ports that have previously generated persistent reservations can be obtained from the logical unit to allow each initiator to monitor which initiator ports have participated in the sharing process.

The programming conventions typically used with RESERVE and RELEASE may conflict with the programming conventions that are used with PROUT and PRIN. Operating systems should only use one of the two reservation command sets at a time with a logical unit.

Modifications required to SPC to implement persistent reservation:

1) Provide additional description in section 5.3 of SPC

Section 5.3 and 5.3.1 should be rewritten as follows:

5.3 Reservations

Various types of reservation commands can be used to prohibit or restrict the execution of certain commands to a logical unit or a portion of the logical unit. Using these reservation commands, application clients can cooperate to protect shared data from accidental modification. If the application clients do not cooperate in the execution of a reservation protocol, data may be unexpectedly modified and deadlock conditions may occur.

Two types of reservation commands are defined. The nonpersistent reservation 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. The reservations created by such commands include reservations for the entire logical unit or for an extent of the logical unit. Extent reservations may place restrictions only upon certain types of commands. The reservations may also be made restricting access to the device to a different initiator, usually a temporary initiator performing a service for the reserving initiator. The reservations do not persist across some recovery actions, so most systems using nonpersistent reservations require significant reinitialization after a failure. Reservations are retained by the logical unit until released or until reset by mechanisms specified in this standard.

The persistent reservation commands, PERSISTENT RESERVE IN and PERSISTENT RESERVE OUT, are used among multiple initiators that do require operations to be protected across initiator failures. The reservations created by such commands include reservations for the entire logical unit or for an extent of the logical unit. Both extent and logical unit reservations may place restrictions only on certain types of commands. The reservations do persist across recovery actions, so that recovery can be managed without requiring complete reinitialization of the system. Reservations for failing initiators can be preempted by an initiator as part of the recovery process. Reservations are retained by the logical unit until released or until reset by mechanisms specified in this standard.

Because a device server cannot differentiate among 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 shall coordinate reservations.

The clause defining each command’s operations shall contain a description of how that command is affected by reservations. The command may be allowed to execute or it may be prevented from execution. If the command is prevented from execution, the command is said to conflict with the reservation and the logical unit shall present a status of RESERVATION CONFLICT. Commands that read or write to the storage medium or to storage caches shall obey the rules defined in the clauses describing the RESERVE command and the PERSISTENT RESERVE OUT command. Commands that retrieve or alter information about the device server’s operating state shall conflict with logical unit reservations unless otherwise specified. Commands that alter information about the device server’s operating state shall conflict, unless otherwise specified, with extent reservations unless the logical unit maintains separate state information for each initiator.

The INQUIRY and REQUEST SENSE commands shall not be affected by any kind of reservation. The RESERVE(6) and RESERVE(10) commands allow superceding reservations and shall be executed even when nonpersistent reservations are present from the same initiator. The RELEASE(6) and RELEASE(10) commands shall be executed for the reserving initiator even when nonpersistent reservations are present.

The execution of a RESERVE(6) or a RESERVE(10) command conflicts with a persistent reservation. The execution of a PERSISTENT RESERVE OUT command with a Reserve, Release, Preempt, or Preempt and Clear action conflicts with a nonpersistent reservation.

2) Replace section 5.4 of SPC concerning Dual Port behavior

The entire clause entitled Dual Port behavior should be replaced with the following text:

5.4 Multiple port and multiple initiator behavior

The SCSI Architectural Model, X3.XXX-19xx, specifies the behavior of logical units being accessed by more than one initiator. Additional ports to a logical unit provide alternate delivery paths through which the device server can be reached and may also provide connectivity for additional initiators. An alternate path can be used to improve the availability of drives in the presence of certain types of failures and to improve the performance of drives whose other paths may be busy.

If a logical unit has more than one SCSI interface port, the arbitration and connection management among the ports is implementation dependent. Two contention resolution options exist:

1) If one port to a logical unit is being used by an initiator, accesses attempted through another port may receive a status of BUSY.
2) If the logical unit has sufficient internal resources, the logical unit may accept actions through other ports while one port is being used.

The device server shall indicate the presence of multiple ports by setting the DualP bit to *1* in its standard INQUIRY data.

From an initiator, each other initiator attached to the logical unit has the same relationship to the logical unit, whether the other initiator accesses the logical unit through the same or a different port. Reservations, persistent reservations, and task management functions are performed between a single initiator and a single logical unit. The following operations are the only operations that allow an initiator to interact with the tasks of another initiator:

PERSISTENT RESERVE OUT with Preempt action removes persistent reservations for another initiator.

PERSISTENT RESERVE OUT with Preempt and Clear action removes persistent reservations and all tasks for another initiator.

Task Management function of TARGET RESET removes nonpersistent reservations and removes all tasks for all initiators. Persistent reservations remain unmodified.

Task Management function of CLEAR TASK SET removes all tasks for all initiators. Most other machine states remain unmodified, including MODE SELECT PARAMETERS, persistent and nonpersistent reservations, and auto contingent allegiance.

3) Remove reference to the Port Status command in Table 5

At present, the referenced section entitled “Port Status” is not included in revision 6 of SPC. It should continue to be not included and should be removed from table 5, page 16.

4) Include PROUT and PRIN commands in SPC:

The PERSISTENT RESERVE OUT command and the PERSISTENT RESERVE IN command will be included in table 5 as “Z” commands.

The following text will be added to SPC:

N.n PERSISTENT RESERVE OUT command

The PERSISTENT RESERVE OUT or PROUT command (see Table 1) is used to reserve a logical unit or an extent within a logical unit for the exclusive or shared use of a particular initiator. The command is used in conjunction with the PERSISTENT RESERVE IN command and cannot be used with the RESERVE and RELEASE commands. Persistent reservations conflict with reservations established by the RESERVE command. Initiators performing PROUT actions are identified by a reservation key provided by the application client. An application client can use the PRIN command to identify which applications are holding conflicting or invalid reservations and use the PROUT command to preempt those reservations if required.

The PROUT and PRIN commands provide the basic mechanism for dynamic contention resolution in multiple-initiator systems using multiple port targets. The identification of reservations using the reservation key makes it possible to determine which ports hold conflicting reservations and to take over reservations from failing or uncooperative initiators.

Table 1: PERSISTENT RESERVE OUT command

Bits / Bytes	7	6	5	4	3	2	1	0
0	Operation Code (TBD)							
1	reserved			Action				
2	Scope				Type			
3	reserved							
4	reserved							
5	reserved							
6	reserved							
7	(MSB) Parameter Length							
8	(LSB)							
9	Control							

The PROUT command contains fields that specify a persistent reservation action, the intended scope of the reservation, and the restrictions caused by the reservation. Parameters contained in the PROUT Parameters specify the reservation keys and extent information required to perform a particular persistent reservation action. The parameters are 24 bytes in length and the Parameter Length field is required to have a value of 24. If the Parameter Length is not 24, a CHECK CONDITION is indicated with an ASC/ASCQ of Invalid CDB. Since persistent reservations are not reset by Target Reset or other global actions, they can be used to enforce device sharing among multiple initiators.

Commands from any initiator that conflict with a successfully established persistent reservation are rejected with a status of RESERVATION CONFLICT. The following commands never conflict with a persistent reservation:

INQUIRY

REQUEST SENSE

PREVENT ALLOW MEDIUM REMOVAL (with a prevent bit of one)

PERSISTENT RESERVE IN

PERSISTENT RESERVE OUT (with an action of Preempt)

PERSISTENT RESERVE OUT (with an action of Preempt and Clear)

PERSISTENT RESERVE OUT (with a reservation action that does not conflict with established persistent reservations or tasks)

Other commands conflict if they perform an operation to the logical unit that violates either the scope or the type specified for an active persistent reservation. Each command of a set of linked commands is individually examined for conflicts at the time the command is received by the target.

N.n.1 PROUT Reservation Actions:

The PROUT command actions are shown in Table 2. The parameters required for each action are shown in Table 6.

Table 2: PERSISTENT RESERVE OUT Action

Action Code (Hex)	Action Name	Action Description
00	Register	Register Reservation Key with target
01	Reserve	Create Persistent Reservation using Reservation Key
02	Release	Release Persistent Reservation
03	(reserved)	
04	Preempt	Pre-empt Persistent Reservation for other initiator
05	Pre-empt and Clear	Persistent Reservation and Clear Task Set for preempted initiator
06-1F	(reserved)	

N.n.1.1 Register

The PROUT command executing a Register action registers a reservation key with a target without generating a reservation. For each initiator that performs a PROUT Register action, the target retains the reservation key until the key is changed by a nonconflicting PROUT command from the same initiator that uses a different key or until the key is reset by powering down the target. The Register action can be performed regardless of any active persistent reservations. All existing persistent reservations for the initiator receive the new reservation key.

N.n.1.2 Reserve

The PROUT command performing a Reserve action creates a persistent reservation having a specified scope and type. The scope and type of a persistent reservation are defined below. For each initiator that performs a PROUT Reserve action, the target retains the reservation key until the key is changed by a PROUT command from the same initiator that uses a different key, until the key is reset by powering down the target, or until the key is cleared by a Clear Key action.

A status of RESERVATION CONFLICT shall be generated for a PROUT command that specifies the execution of a Reserve action that conflicts with any active persistent reservations from the same initiator in scope, type, extent, or reservation key.

Persistent reservations shall not be superceded by a new persistent reservation from any initiator except by execution of a PROUT specifying either the Preempt or Preempt and Clear action. New persistent reservations that do not conflict with an existing persistent reservation shall be executed normally. The reservation of a logical unit or the reservation of reserved extents having the same type value is permitted if no conflicting persistent reservations are held by another initiator. When such overlapping reservations are released, each of the extent reservations and the logical unit reservation shall be removed with a separate Release action.

A PROUT command not performing a Preempt or a Preempt and Clear action shall not be performed and shall be ended with status of RESERVATION CONFLICT if there are any queued or active tasks from any initiator that would conflict with the reservation to be established.

N.n.1.3 Release

The PROUT command performing a Release action removes an active persistent reservation held by the same initiator. The parameters associated with the Release action must match the parameters of the active reservation. It is not an error to send a PROUT specifying a Release action for a persistent reservation that does not exist. The reservation key is not changed by the Release action.

A status of CHECK CONDITION and an ASC/ASCQ of Invalid Release of Active Persistent Reservation shall be generated for a PROUT command that specifies the release of a persistent reservation with an incorrect scope, reservation key, or extent.

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

1) Power off.
2) Execution of a PROUT command from another initiator with a Persistent Reserve Action of Preempt or Preempt and Clear

N.n.1.4 Preempt

The PROUT command performing a Preempt action removes all reservations for the initiator specified by the PROUT parameter page. The initiator is identified by the reservation key of the initiator to be preempted. Any commands from any initiator that have been accepted by the logical unit as nonconflicting will continue normal execution.

A Unit Attention condition is established for the preempted initiator. The first new command from the preempted initiator shall present CHECK CONDITION and present the Unit Attention condition with an ASC/ASCQ of Unit Attention/Reservations preempted. Subsequent commands are subject to the reservation restrictions established by the preempting initiator.

The persistent reservation created by the preempting initiator is specified by the scope and type field of the PROUT command and the corresponding parameters in the PROUT parameter page.

N.n.1.5 Preempt and Clear

The PROUT command performing a Preempt and Clear action removes all reservations for the initiator specified by the PROUT parameter page. The initiator is identified by the reservation key of the initiator to be preempted. Any commands from the initiator being preempted are each terminated as if an ABORT TASK task management function had been performed by the preempted initiator.

A Unit Attention condition is established for the preempted initiator. The first new command from the preempted initiator shall present CHECK CONDITION and present the Unit Attention condition with an ASC/ASCQ of Unit Attention/Reservations preempted. Subsequent new commands and retries of commands that timed out because they were cleared are subject to the reservation restrictions established by the preempting initiator.

The persistent reservation created by the preempting initiator is specified by the scope and type field of the PROUT command and the corresponding parameters in the PROUT parameter page.

N.n.2 PERSISTENT RESERVE OUT Scope

The value in the Scope field indicates whether a persistent reservation applies to an entire logical unit or to a portion of the logical unit defined as an extent. The values of the Scope field are defined in Table 3.

Table 3: PERSISTENT RESERVE OUT Scope

Code (Hex)	Scope Name	Scope Description
0	LU	The Persistent Reservation Out is applied to the full logical unit.
1	Extent	The Persistent Reservation Out is applied to the specified extent.
2-F	(reserved)	

N.n.2.1 LU

A Scope field value of LU indicates that the persistent reservation applies to the entire logical unit.

N.n.2.2 Extent

A Scope field value of Extent indicates that the persistent reservation applies to the extent of the logical unit defined by the extent parameters in the PROUT parameter page. An extent is defined only for devices defining contiguous logical block addresses.

N.n.3 PERSISTENT RESERVE OUT Type

The value in the Type field specifies the characteristics of the persistent reservation being established for all data blocks within the extent or within the logical unit. Table 4 describes the characteristics of the five different type values.

Table 4PERSISTENT RESERVE OUT Type

Code (Hex)	Type Name	Type Description		
		Read	Write	Permitted Reservations
0	Read Shared	Shared	Prohibited	Nonconflicting, any initiator
1	Write Exclusive	Shared	Exclusive	Nonconflicting, any initiator
2	Read Exclusive	Exclusive	Shared	Nonconflicting, any initiator
3	Exclusive Access	Exclusive	Exclusive	Nonconflicting, this initiator
4	Shared Access	Shared	Shared	Nonconflicting, this initiator
5-F	(reserved)			

N.n.3.1 Read

If a type allows read commands to be shared, then any initiator can execute commands that perform transfers from the storage medium or cache of the logical unit to the initiator. If a type requires read commands to be exclusive, then only the reserving initiator can perform a transfer from the storage medium or cache to the initiator. Any SCSI commands from another initiator that would create such a transfer will receive status of RESERVATION CONFLICT.

N.n.3.2 Write

If a type allows write commands to be shared, then any initiator can execute commands that perform transfers from the initiator to the storage medium or cache of the logical unit. If a type requires write commands to be exclusive, then only the reserving initiator can perform a transfer from the initiator to the storage medium or cache of the logical unit. Any SCSI commands from another initiator that would create such a transfer will receive status of RESERVATION CONFLICT. If a type requires write commands to be prohibited, then any command from any initiator that would create such a transfer will receive status of RESERVATION CONFLICT.

N.n.3.3 Permitted Reservations

If a type allows nonconflicting reservations from any initiator, then any initiator may generate persistent reservations to the same extent or logical unit as long as the characteristics are not contrary to the existing reservations. Attempts to create conflicting reservations will receive status of RESERVATION CONFLICT.

If a type allows nonconflicting reservations only from the reserving initiator, then all attempts by other initiators to perform any PROUT command with the Reserve action will receive status of RESERVATION CONFLICT. Attempts to create conflicting reservations by the reserving initiator will receive status of RESERVATION CONFLICT.

N.n.4 PERSISTENT RESERVE OUT parameters

The parameters required to perform the PROUT command are defined in Table 5 and in Table 7. Table 6 indicates which parameters are required for which Action and Scope value. The locations for all parameter fields are always required to be transmitted, even if the parameter is not required for the specified function

Table 5: PERSISTENT RESERVE OUT parameters

Bit	Function of Byte							
Byte	7	6	5	4	3	2	1	0
0 through 23	PROUT parameter page							

Table 6: PERSISTENT RESERVE OUT: Allowed actions and valid parameters

Action	Allowed Scope	Parameters Valid		
		Key for port performing PROUT	Key for port preempted by PROUT	Extent LBA and Extent Length
Register	LU	Valid	N/A	N/A
Reserve	LU	Valid	N/A	N/A
Reserve	Extent	Valid	N/A	Valid
Release	LU	Valid	N/A	N/A
Release	Extent	Valid	N/A	Valid
Preempt	LU	Valid	Valid	N/A
Preempt	Extent	Valid	Valid	Valid
Preempt and Clear	LU	Valid	Valid	N/A
Preempt and Clear	Extent	Valid	Valid	Valid

N.n.4.1 Reservation Key of initiator performing command

The reservation key is an 8-byte token provided by the initiator to the logical unit to identify the source of the PROUT command.

N.n.4.2 Reservation Key of initiator being preempted

The reservation key of the initiator being preempted is only required for those PROUT actions that preempt another initiator’s reservations. This reservation key matches the reservation key of the persistent reservation that is being preempted. If it does not match, the PROUT command presents status of RESERVATION CONFLICT.

N.n.4.3 Extent definition parameters

The extent parameters are only required for those PROUT actions that use a Scope value of Extent. The extent is defined by the 32-bit starting logical block address and the 16-bit count of logical blocks in the extent.

Table 7: PERSISTENT RESERVE OUT parameter page

	Function of Byte							
Bit Byte	7	6	5	4	3	2	1	0
0 through 7	(MSB) Reservation Key of initiator performing command (LSB)							
8 through 15	(MSB) Reservation Key of initiator being preempted (LSB)							
16 -19	(MSB) LBA of first block of extent (LSB)							
20	(reserved)							
21	(reserved)							
22 - 23	(MSB) Extent Length (MSB)							

N.m PERSISTENT RESERVE IN command

The PERSISTENT RESERVE IN or PRIN command (see Table 8) is used to obtain information about reservations and reservation keys that are active within a logical unit. The command is used in conjunction with the PERSISTENT RESERVE OUT command and cannot be used with the RESERVE and RELEASE commands.

The actual length of the parameters that could be returned by the PRIN command is defined within the parameter list. The Parameter 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 is returned. If the remainder of the list is required, a new PRIN command with a Parameter Length field large enough to contain the entire list is executed.

Table 8: PERSISTENT RESERVE IN command

Bits / Bytes	7	6	5	4	3	2	1	0
0	Operation Code (TBD)							
1	reserved			Action				
2	reserved							
3	reserved							
4	reserved							
5	reserved							
6	reserved							
7	(MSB) Parameter Length							
8								(LSB)
9	Control							

N.m.1 PERSISTENT RESERVE IN action codes

The action codes for the PRIN command are defined in Table 9.

Table 9: PERSISTENT RESERVE IN Action

Action Code (Hex)	Action Name	Action Description
00	Read Keys	Reads all registered Reservation Keys
01	Read Reservations	Reads all current reservations
02-1F	(reserved)	(reserved)

N.m.1.1 Read Keys

The Read Keys action value requests that the logical unit return a parameter list containing a header and a complete list of all reservation keys that have been passed to the logical unit from all initiators. The keys may have been passed by a PROUT command that has performed a reserve action, a register action, or one of the preempt actions. The reservation keys do not indicate what initiator or port is associated with the key. That information must be obtained from other initiators by mechanisms outside the scope of this standard.

N.m.1.2 Read Reservations

The Read Reservations action value requests that the logical unit return a parameter list containing a header and a complete list of all reservations that are presently active on the logical unit and its extents.

N.m.2 PERSISTENT RESERVE IN parameters for Read Keys

The format for the parameters provided in response to a PRIN command with the Read Keys action is shown in Table 10.

Table 10: PERSISTENT RESERVE IN parameters for Read Keys

	Function of Byte							
Bit / Byte	7	6	5	4	3	2	1	0
0 through 3	(MSB) Generation (LSB)							
4 through 7	(MSB) Additional length of parameter field (LSB)							
8 through 15	(MSB) Zeroth Reservation Key (LSB)							
8+8n through 15+8n	(MSB) Nth Reservation Key (LSB)							

N.m.2.1 Generation

The Generation value is a 32-bit counter in the logical unit that is incremented every time a PROUT command requests a Reserve, a Preempt, or a Preempt and Clear operation. The counter is not incremented by a PRIN command, by a PROUT command that performs a Register or Release action, or by a PROUT command that is not performed due to an error or reservation conflict.

The Generation value allows an initiator to verify that reservations have not been updated in the logical unit without the knowledge of the initiator.

N.m.2.2 Additional Length

This field contains a count of the number of bytes in the reservation key list. The total number of bytes available in a particular PRIN command is 8 more than the value in the

Additional Length field. If the Parameter Length specified by the PRIN command is not sufficient to contain the entire parameter list, then only the bytes from 0 to the maximum allowed Parameter Length are transmitted to the initiator. The remaining bytes are truncated, although the Additional Length field still contains the number of bytes in the reservation key list that would have been transmitted. In the case of such a truncation, CHECK CONDITION status with sense information of incorrect length is provided.

N.m.2.3 Reservation Key list

This list contains all the 8-byte reservation keys known to the logical unit through PROUT Reserve, Preempt, Preempt and Clear, or Register actions. Each reservation key may be examined by the host system and correlated with a particular initiator and SCSI port by mechanisms outside the scope of this standard.

N.m.3 PERSISTENT RESERVE IN parameters for Read Reservations

The format for the parameters provided in response to a PRIN command with the Read Reservations action is shown in Table 11.

Table 11: PERSISTENT RESERVE IN parameters for Read Reservations

	Function of Byte							
Bit Byte	7	6	5	4	3	2	1	0
0 through 3	(MSB) Generation (LSB)							
4 through 7	(MSB) Additional length of parameter field (LSB)							
8 through 23	(MSB) Zeroth Read Reservation Page (LSB)							
8+16n through 23+16n	(MSB) Nth Read Reservation Page (LSB)							

The Generation field and Additional Length fields are defined exactly the same as the fields for the parameters for the Read Keys action. The read reservation pages are defined in Table 12. There is one read reservation page for each reservation held on the logical unit by any initiator.

Table 12: PERSISTENT RESERVE IN Read Reservation Page

	Function of Byte							
Bit / Byte	7	6	5	4	3	2	1	0
0 through 7	(MSB) Reservation Key of port holding reservation (LSB)							
8 through 11	(MSB) LBA of first block of Extent (LSB)							
12	(reserved)							
13	Scope				Type			
14 15	(MSB) Extent Length (MSB)							

N.m.3.1 Read Reservation Page parameters

For each reservation held on the logical unit, there shall be a Read Reservation page presented in the list of parameters returned by the logical unit to the PRIN command with a Read Reservations action. The page contains the reservation key of the initiator holding the reservation. The type and scope of the reservation are also defined. If the scope is an Extent reservation, then the LBA of the first block of the extent and the extent length parameters are valid. If the scope is a Logical Unit reservation, then bytes 8 through 11, 14, and 15 are set to 0.

New ASC/ASCQ definitions required:

The following new ASC/ASCQ indications are required to present error information associated with persistent reservations:

Invalid Release of Active Persistent Reservation

Invalid Parameter (offered when more than one parameter page is specified by PROUT Preempt).

Unit Attention/Reservations Preempted

Incorrect Length (offered when Parameter Length is too short for PRIN parameter lists.)

Invalid CDB (offered when Parameter Length is not = 24 in PROUT)

Proposed text changes to SBC, SSC, and SCC

The PERSISTENT RESERVE IN and PERSISTENT RESERVE OUT commands shall be included in the introductory command maps for all three command sets.

Those commands and the RESERVE(6), RESERVE(10), RELEASE(6), and RELEASE(10) shall be marked as X (for exclusive mandatory) instead of M. A note shall indicate that at least one of the following sets of commands is mandatory:

RESERVE(6), RELEASE(6)

PERSISTENT RESERVE IN, PERSISTENT RESERVE OUT

The note shall further indicate that technologies using extended logical unit addresses shall implement at least one of the following sets of commands as mandatory:

RESERVE(10), RELEASE(10)

PERSISTENT RESERVE IN, PERSISTENT RESERVE OUT

Proposed text changes to other SCSI-3 documents:

None.

Arbitrary decisions made developing this document, please review:

1) Single key per initiator

The definition of persistent reservation at present uses a single key for each initiator. The definition could use a single key for each reservation, but that would significantly increase the space required for keys in the logical unit. The use of multiple keys for an initiator holding multiple persistent reservations does not appear to add any useful function.

2) Modification of key

The key is presently specified as being modifiable during persistent reservations by the Register action. The key is not modifiable during the execution of additional reservations or overlapping reservations to a logical unit. Invalid keys during Reserve actions are RESERVATION CONFLICTS.

3) Management of queued tasks

Queued tasks from any initiator continue normally if a PROUT with a Preempt command is executed. The assumption is that they have been accepted as nonconflicting and if you really wanted to kill them off, you would do so with a Preempt and Clear.

However, a Reserve action can only be performed if there are no conflicting queued tasks. The assumption is that the initiators are cooperating and have agreed upon the reservations to be performed in such a manner that any conflicts are errors. Note that this may require targets to do a lot of task inspection to see if a conflict exists.

For a Preempt and Clear command, only those queued tasks for the specified initiator are cleared. Other queued tasks complete normally. Note that this allows the creation of conflicting tasks that will be completed normally, unlike the Reserve action. No new conflicting tasks are accepted.

4 Command linking

It is assumed that each command of a linked set of commands will be checked for conflicts independently, in spite of the fact that they are nominally all part of the same task. This is necessary because the initial command may be nonconflicting, but subsequent commands conflicting. This is also true for the completion of commands that have not yet been queued but are part of an on going task during preemption.

5) Number of reservations per PROUT

At present, the document specifies one reservation per PROUT. This is a requirement for all commands except PROUT with the Reserve action, so it seemed natural to make the same restriction for that case as well. Theoretically, multiple extents could be reserved at the same time, but the Scope and Type parameters are presently in the command, so all reservations would have to be identical except for extent. In addition, all reservation keys would have to be identical. For now, only one reservation at a time can be performed.

6) Reservation established by preempting initiator

There is no mechanism to preempt an initiator’s reservations without establishing a reservation of some kind with the preempting initiator. A null reservation scope could be established to allow preemption without such a new reservation, but this does not seem to be a useful function.

7) Target Reset

As presently defined in SAM, TARGET RESET and CLEAR TASK SET clear all tasks for all initiators. This proposal retains that property and includes that definition in the proposed modifications to clause 5.4 of SPC. Some proposals have suggested that TARGET RESET should only reset the tasks for the initiator that generated the task management function. Unless the committee directs otherwise, I propose that we leave the definitions unchanged and accept the clear understanding that TARGET RESET will mess up all initiators and should not be used. At least it will not clear persistent reservations.

8) Reservation Conflicts

At present, it appears that a RESERVE(6) or RESERVE(10) command does not conflict with a reservation from another initiator. I believe it should be required to conflict for all logical unit reservations by another initiator and should be required to conflict for overlapping and inconsistent extent reservations by another initiator. This will require modification to clause 5.3 and to the definitions of the RESERVE commands.