Date: February 23, 2002
To: T10 Committee (SCSI)
From: Mallikarjun Chadalapaka (cbm@rose.hp.com) and Randy Haagens
(randy_haagens@hp.com), Hewlett-Packard.
Subject: Reservations & Nexus

1. Introduction and history

FCP-2 specifies that the Reserve-Release managed reservations (hereinafter referred to as "regular reservations") are released when the PRLI session object is implicitly or explicitly destroyed. iSCSI at the moment (as of rev11) takes its cue from FCP-2, largely to make it easier for the iSCSI-to-FCP bridges, in specifying that regular reservations are cleared when the iSCSI Login session is implicitly or explicitly destroyed. It is also to be noted that parallel SCSI (hereinafter referred to as "pSCSI") does not clear the reservations when the I_T nexus (roughly equivalent to FCP-2 and iSCSI login sessions) object is destroyed. SRP is currently considering if reservations should be retained across SRP logouts - i.e. generally similar to pSCSI's behavior.

The purpose of this memo is to explore the reasons and options behind this divergence between transports and to seek architectural guidance from the T10 committee. This will assist all SCSI transport protocols to make considered design decisions consistent with SCSI architecture.

The authors gratefully acknowledge the comments from Marjorie Krueger, George Penokie, Ralph Weber and Rob Elliott which helped frame the discussion presented here.

2. Why this divergence?

The authors believe that the following factors contributed to the aforementioned divergence in the way different transport protocols defined the life of regular reservations.

- Networked storage transports like FCP-2 (and iSCSI for now) wanted to directly associate the life of a regular reservation to that of an I_T nexus, and defined the I_T nexus as the protocol "session" (process login session for FCP-2). This desire is based on the rationale that an initiator that "disappears" from a storage fabric may or may not ever come back to reclaim (and release) the reservation. These transports did not want the SCSI targets to continue to hold the reservations forever - thus eventually forcing an initiator wanting access to use a target/LU reset to clear the reservations, the consequences of which are likely to be severe on several other initiators in a networked storage environment. [*However, this desire unfortunately raises some serious concerns about the efficacy of reservations should the transport allow the reservation to disappear when none of the cluster principals request releasing the reservation - i.e. an "implicit" release due to external factors. As far as the authors can tell, both FCP-2 and iSCSI allow this.*]

- SAM-2, rev 21, clause 4.10, defines a nexus object, and details the different kinds of nexuses that are permissible: I_T, I_T_L, I_T_L_Q, or I_T_L_x nexus. But it *does not* define
  a. the duration of validity (or life) of the objects, more specifically when each of these objects comes into existence and when it is destroyed.
  b. the inter-dependencies (or lack thereof) among these objects (for ex., can I_T_L nexus object exist in the absence of the corresponding I_T nexus object?)

- Neither SPC-3 nor SPC-2 specifically associates the life of a reservation to the duration of the corresponding I_T_L nexus object, though one may argue that the association is implicit. If it *were* explicit - since the transport protocols do not concern themselves with ULP-abstractions such as LUNs (other than carrying a LUN in the transport envelope) - it might have motivated FCP-2 (and thus iSCSI) *not* to require the reservations to be cleared on the disappearance of an I-T nexus.

3. Regardless of which option we pick...

Before we launch into a detailed analysis of a few design options, let us review the underlying architectural assumptions that all the presented options presuppose.

- The I_T_L nexus object is completely in the domain of the SCSI ULP layer. What this means is that the ULP owns the nexus object and the object (at least logically) exists at the ULP level regardless of the transport. This needs to be so specified in SAM-2. (*Each of the design options presented in section 4 define the duration of I_T_L nexus object for that case.*)

- The regular reservation object is is completely in the domain of the SCSI ULP layer.

- The I_T_L_Q nexus object is instantiated when the corresponding I_T_L nexus object is also instantiated and when a task with a tag Q is issued on the nexus. The I_T_L_Q nexus object is destroyed on the conclusion of the said task, or when the I_T_L nexus object is destroyed. This needs to be so specified in SAM-2.

None of the deployed SCSI implementations needs to change regardless of the option we choose, but SCSI transport protocol documents would need to appropriately reflect the option we choose here. The purpose of this proposal is to define a sound architectural basis for future transports, but *not to* require any implementation changes/upgrades.

4. What are the Design options?

4.1. *Option A*: Reservations have nothing to do with transport layer dynamics

This option argues that from a clean layering perspective, all regular reservations are abstractions known only to the SCSI layer by definition and hence are unaffected by transport level dynamics, which happen at a lower level. When an I_T nexus is re-established between the same SCSI ports, the pre-existing I_T_L nexus objects and reservation objects are implicitly associated with the new instantiation of the nexus.

In this model, I_T nexus object is completely contained in the transport layer and is completely abstracted from the SCSI ULP.

Pros -
- Enforces clean layering, transport layer doesn't know or care about ULP abstractions.
- Relieves the transport protocols from specifying anything in relation to reservations.
- Substantiates the behavior of pSCSI.
- Good for multi-protocol bridges.

Cons -

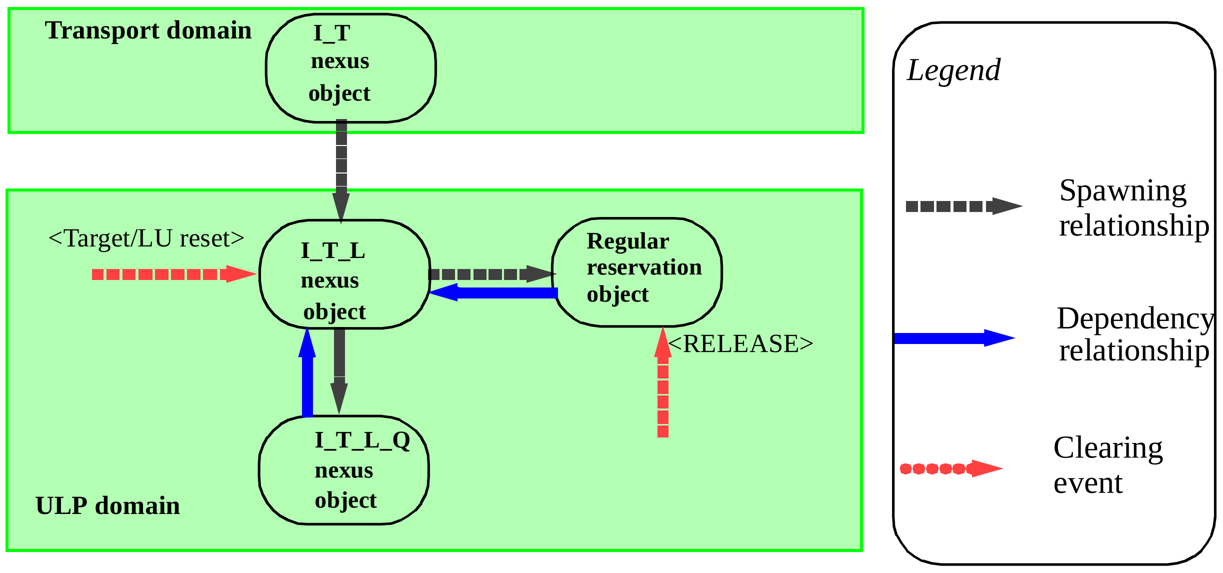
- Requires an exception for FCP-2's reservation behavior.
- Does not address the requirement that all tasks be terminated deterministically on both ends when the "login session" collapses for networked transports.
- Requires targets in networked transports to maintain reservation state indefinitely for disappeared initiators. [Sidenote: This issue can be addressed in one of two ways: *(a)* define a new SCSI command that clears all third-party regular reservations (sort of a light-weight LU reset), or *(b)* define a scheme of leases for reservations so they automatically expire.]

4.1.1.Life of I_T_L nexus and reservation objects

In this model, the I_T_L nexus object is instantiated when the first valid task to the LU is received and accepted (i.e. the task enters the Dormant state) and it is destroyed when the target/LU is reset.

The reservation object is instantiated on RESERVE, and destroyed on RELEASE, or upon destruction of the I_T_L nexus object.

4.1.2.Object relationship diagram



4.2. *Option B*: Transport influences when I_T_L nexus goes away, and also reservations

This school of thought argues that we need to:

a. make it explicit in SAM-2 that the SCSI ULP has a notion of "logical" I_T nexus. In other words, make the architectural statement that SCSI ULP is a connection-oriented protocol.
b. allow each transport to specify when to notify the SCSI ULP about a solicited/unsolicited destruction of "logical" I_T nexus object. This in turn causes the SCSI ULP to discard all the associated I_T_L nexus objects (even though the object was instantiated upon ULP actions). SAM-2 defines a new protocol notification service "LostNexus" to be communicated across the protcol service interface from the transport layer into the SCSI ULP layer.

c. architect a new protocol service notification "LostReservations" to be communicated across the protcol service interface to be communicated from the transport layer into the SCSI ULP layer.

At first glance, this option seemingly presents a fundamental problem for pSCSI since the term "I_T nexus" is traditionally used to represent the physical bus nexus, and clearly the reservations are not discarded on every BUS FREE! But consider the following rationale:
When the physical bus nexus is not present (i.e. on BUS FREE), pSCSI already supports the notion of

- a logical I_T_L_Q nexus object (continuing tasks across disconnects and reconnects)
- a logical I_T_L nexus object (supporting reservations even on a quiesced bus)

It appears merely a mental experiment to extend the same "logical-ness" to I_T nexus object. Why not extend?

To summarize, in this model, the I_T_L nexus object is dependent on the existence of I_T nexus object (in the ULP domain), and transport dictates the duration of I_T nexus object.

Pros -

- This acknowledges the connection-oriented nature of several new SCSI transports, by abstracting that nature into SCSI ULP. A non-networked transport is merely a degenerate case.
- This allows each transport to specify when to generate the LostNexus and LostReservations notifications across the protocol service interface.
- The delinking of reservation object from I_T_L nexus object allows frivolous storage network transport conditions from clearing the reservations (for ex., iSCSI TCP connection failure), even while it clears I_T and I_T_L nexus objects.
- This requires no exceptions for pSCSI even while accommodating FCP-2.
- This creates the architectural basis for FCP-2's and iSCSI's requirement to terminate the active tasks on "session failure".
- This option provides the architectural basis for the AccessID enrollment state to be associated to the I_T nexus object (since both are in the ULP domain).

Cons -

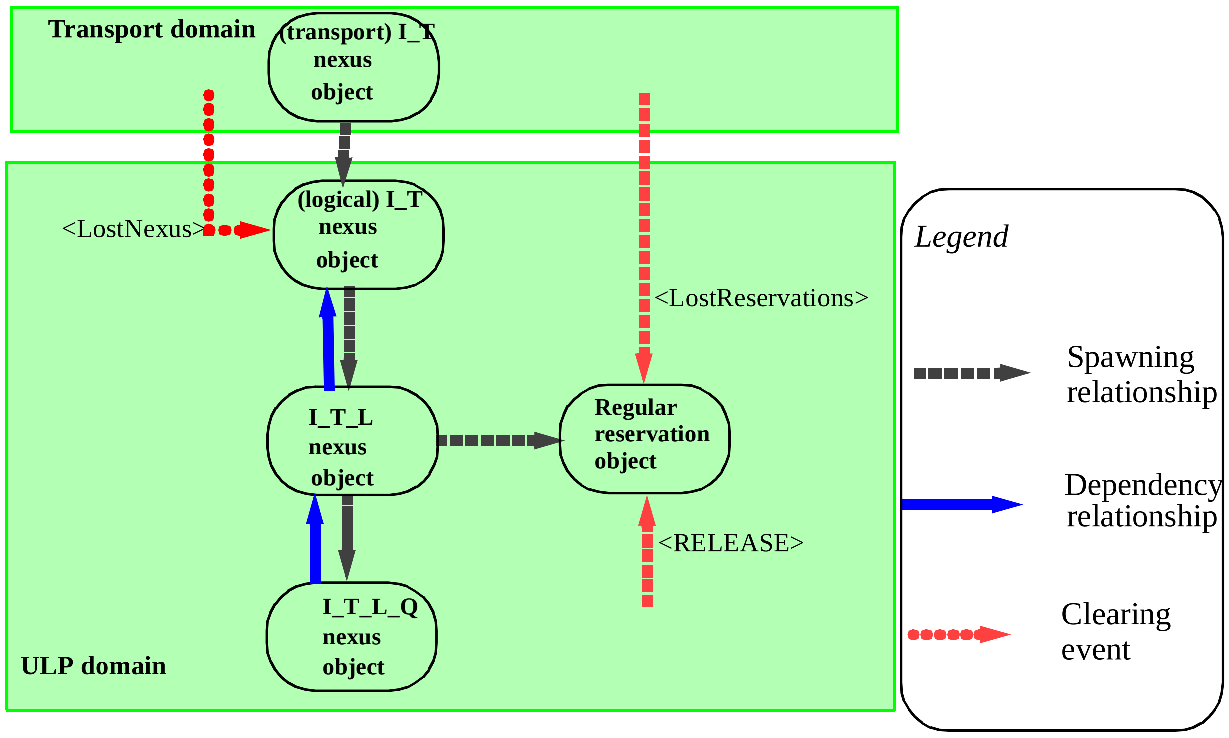
- This introduces an asymmetry between the creation and destruction of the I_T_L nexus object. While SCSI ULP always causes the instantiation of the object, either the transport or the ULP (as in target reset task management function) directs the decision to discard the object.

4.2.1.Life of I_T_L nexus and reservation objects

In this model, the I_T_L nexus object is instantiated when the first valid task to the LU is received and accepted (i.e. the task enters the Dormant state) and destroyed upon LostNexus asynchronous notification to the ULP (about the lost parent I_T nexus).

The reservation object is instantiated on RESERVE, and destroyed on RELEASE, or upon LostReservations asynchronous notification to the ULP.

4.2.2.Object relationship diagram



4.3.*Option C*: Validity of reservations is transport-specific and per LU

This line of reasoning argues that since the regular reservations are SCSI abstractions, it is reasonable to indicate the nature of reservation handling via SCSI ULP means (even while the nature may be protocol-defined). In this model, it is acceptable for regular reservations to be cleared when the I_T nexus is destroyed and this option argues that all networked transports would choose to do it (for the “disappeared initiator” reason). However transports like pSCSI would continue to delink (the physical) I_T nexus from regular reservations.

Rob Elliott presented a scheme to the T10 reflector on 02.21.2002 that generally falls under this category. Rob suggested that a bit in an LU mode page may indicate if reservations are cleared in a protocol-defined way (as in networked transports), or not (as in the case of pSCSI).

Pros -

- Most expedient solution, that essentially makes the architectural statement that the differing behaviors are intended. Each transport protocol defines if it is of “networked” class or not.

Cons -

- This does not provide any architectural guidance to new transports, essentially allows divergence as each sees fit.
- This would allow even unexpected transport exceptions to clear reservations.

- This approach continues to create issues for multi-protocol bridges unless future transports decide to require both “networked” and non-networked behaviors from all LUs (which is unlikely).
- Protocol-specific expectations are imposed all the way upto the LU level.

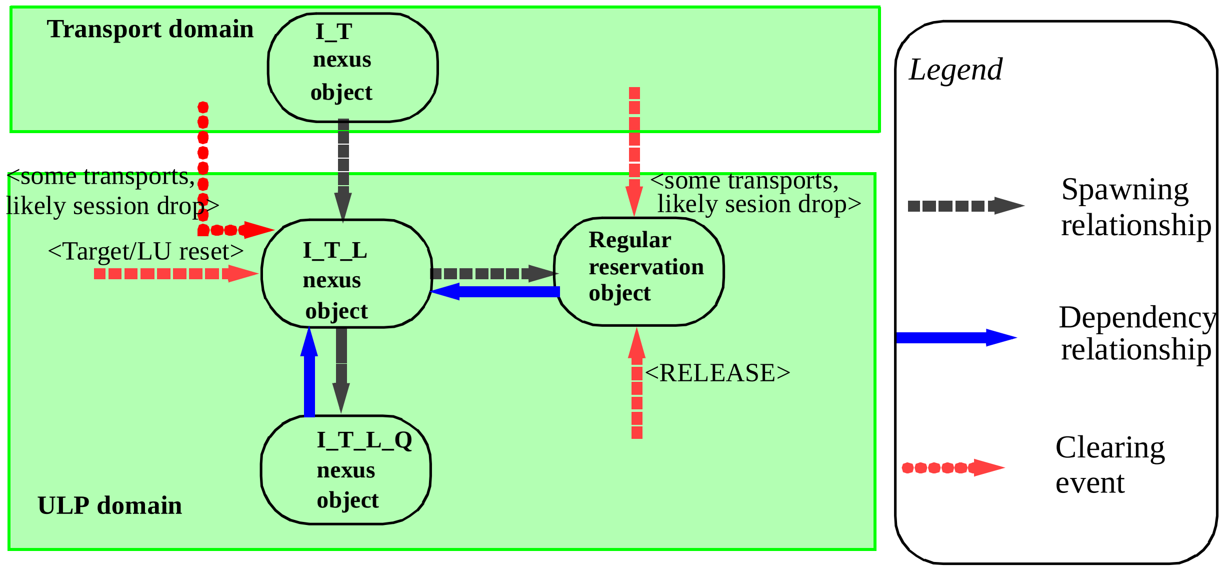
4.3.1.Life of I_T_L nexus and reservation objects

In this model, in the case of networked transport protocols, the I_T_L nexus object is instantiated when the first valid task to the LU is received and accepted (i.e. the task enters the Dormant state) and it is destroyed when the target/LU is reset, or when the I_T nexus object is destroyed.

In this model, in the case of non-networked transport protocols, the lifetime of I_T_L nexus object is as described in section 4.1.1.

The reservation object is instantiated on RESERVE, and destroyed on RELEASE, or upon transport-defined conditions.

4.3.2.Object relationship diagram



4.4.*Option D*: Let’s obsolete reservations!

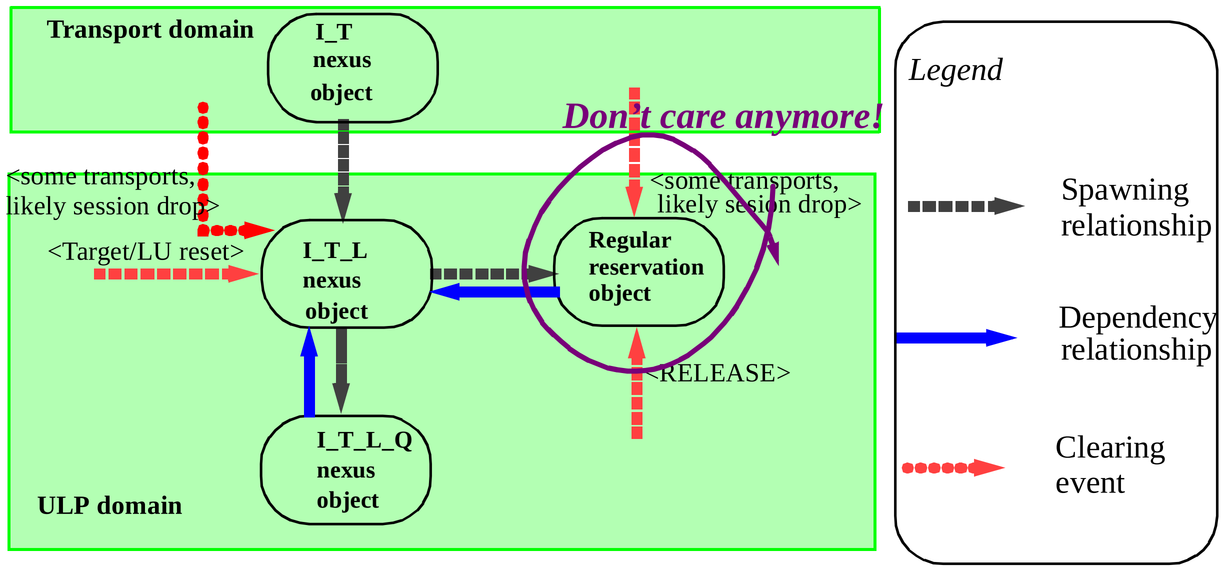
It was pointed out to the authors that regular reservations are not heavily used in practice, and perhaps not the right mechanism to deploy going forward in any case. If deployment and usage of persistent reservations is the desired objective, we can choose to deprecate regular reservations to encourage all SCSI implementations to move away from using them.

Pros -
- May be just the right step from a long-term perspective.
- No need to rationalize the exising contradictions in different transports.
- No need for new transport protocols to address this issue at all.

Cons -
- Forces everyone to transition to a mechanism that is relatively more involved than simple regular reservations.
- Deprecating reservations will not address other architectural issues on hand! Certain changes are called for in any case, this can't be the only element in a solution package.

4.4.1.Life of I_T_L nexus and reservation objects
The existing differences in the life expectancy of I_T_L nexus object will continue until the protocols are obsoleted.

The existing differences in the life expectancy of regular reservation object will continue until the protocols are obsoleted.

4.4.2.Object relationship diagram


5. Now the windfall
While this proposal for the formalization of nexus objects in the SCSI documents was made primarily focusing on Reservations, it turns out that the proposed changes enable other highly desirable changes as well. The authors have identified the following so far.
- SAM-2 currently defines the reset value of CRN (Command Reference Number) to be 1. It should be augmented to state that the CRN is reset when the I_T_L nexus object is instantiated.
- SPC-3 should be modified to state that the mode pages are reset (to the default or to the saved pages) upon the instantiation of the I_T_L nexus object. In particular, the authors propose that "Table 6 - Management of mode pages during PRLI and PRLO" in FCP-2 Revision 7a with minor changes be adopted into SPC-3 (essentially with login and logout wording replaced with I_T_L nexus object instantiation and I_T_L nexus object destruction respectively).

- SAM-2 lists the events that can clear a CA (Contingent Allegiance) and ACA. The list of events for both CA and ACA should be enhanced to include the event “I_T_L Nexus Object destroyed”.
- SAM-2 lists the events that generate a Unit Attention condition. The list of events should include “I_T_L Nexus Object instantiated”.
- If Option B is chosen: SPC-3 should specify that the AccessID enrollment state should be reset to either “pending-enrolled” or “not-enrolled” state (the choice being vendor specific) on the “I-T Nexus Object instantiated” event.

6. Conclusion

It is hoped that this memo will provide a suitable framework for deliberation of various architectural alternatives and help the T10 committee in providing guidance to SCSI transport protocols on the right course of action.

In general, more specific language and a precise definition of lifetime for all nexus objects is necessary in SCSI documents (SAM-2 and/or SPC-3) to achieve the following goals -

- an unambiguous architectural direction for new SCSI transport protocols,
- more precision to the current SCSI architecture and protocol documents,
- less burden on existing and emerging SCSI transport protocols not to specify actions and text that are most appropriate to the SCSI architecture and protocol documents (the PRLI/PRLO table 6 in FCP-2 is an example).
- reaping the additional benefits described in section 5.

The authors would also like to point out that while the proposed list of changes in section 5 may not be exhaustive, they are certainly representative of the type of changes that are to be made to SCSI architecture and command documents.