To: T10 Technical Committee
From: Rob Elliott, HP (elliott@hp.com)
Date: 8 June 2007
Subject: 07-268r0 SAM-4 Protocol services as UML operations

Revision history
Revision 0 (8 June 2007) First revision

Related documents
sam4r11 - SCSI Architecture Model - 4 (SPC-4) revision 11
07-157 - SAM-4 Changes requested at editing session (George Penokie, IBM)
07-263 - SAM-4 SCSI Initiator and Target Port capabilities attributes (Rob Elliott, HP)
UML 2.0 Superstructure Specification

Overview
1. The SCSI transport protocol services are proposed to be mapped to UML operations in this manner:
   1) an application client object asks a SCSI initiator port to send a command by invoking the SCSI initiator port object's Send SCSI Command () operation;
   2) the SCSI target port receives the command and invokes the device server's SCSI Command Received () operation to notify the device server;
   3) the device server asks the SCSI target port to perform the necessary data transfers by invoking the SCSI target port's Send Data-In () and Receive Data-Out () operations;
   4) the SCSI target port informs that device server that each of those is complete by invoking the device server's Data-In Delivered () and Data-Out Received () operations;
   5) the device server asks the SCSI target port to return the status by invoking the SCSI target port's Send Command Complete () operation
   6) When the SCSI initiator port receives the status, it invokes the application client's Command Complete Received () operation to notify the application client.

Each object presents operations that it performs on request by other objects - a SCSI initiator port is the object that actually sends the command on the wire, while an application client is the object that constructs a command and asks a SCSI initiator port to send the command.

Example application client pseudo-code:

```
SCSI_Initiator_Port myInitPort;
...
CDB->OpCode = SCSI_INQUIRY;
myInitPort::Send SCSI Command (&CDB, TA_SIMPLE, &DataInBuffer, NO_BUFFER, CRN_ZERO,
                    TASK_PRIORITY_0, FIRST_BURST_DISABLED);
Wait (Command_Complete_Received_flag);

CDB->OpCode = SCSI_REPORT_LUNS;
myInitPort::Send SCSI Command (&CDB, TA_SIMPLE, &DataInBuffer, NO_BUFFER, CRN_ZERO,
                    TASK_PRIORITY_0, FIRST_BURST_DISABLED);
Wait (Command_Complete_Received_flag);
...
```

Example device server pseudo-code:

```
SCSI_Target_Port myTargPort;
...
Wait (SCSI_Command_Received_flag);
switch (CDB->Opcode) {
case SCSI_INQUIRY: {
        myTargPort::Send Data-In (&InquiryData, 0, sizeof (InquiryData));
        myTargPort::Send Command Complete (&SenseData, STATUS_GOOD,
                    SR_TASK_COMPLETE, RETRY_DELAY_TIMER_ZERO);
```

```
            break;
    }
    case SCSI_REPORT_LUNS: {
            myTargPort::Send Data-In (&ReportLunsData, 0, sizeof (ReportLunsData));
            myTargPort::Send Command Complete (&SenseData, STATUS_GOOD,
                    SR_TASK_COMPLETE, RETRY_DELAY_TIMER_ZERO);
            break;
    }}
    ...
```

2. On the initiator side, changes are made to the task-related objects to parallel the structure of tasks/task sets on the target side:

a) Task -> Application Client Task
b) Task Management Function -> Application Client Task Management Function
c) Application Client Task removed
d) Application Client Task Set - contains 0 or more Application Client Tasks

This avoids having two classes with the same name (Task) but with different meaning to an initiator and target.

“Application client task” was added in a previous revision of SAM-2 to provide nomenclature that is not really necessary with the UML model-based SAM-4. An “application client task set” is helpful for the UML model, representing all the tasks that the application client has outstanding (e.g., for which it is waiting for status).

3. A variety of attributes are added to Application Client Task, Application Client Task Management Function, Task, and Task Management Function representing all the values that are delivered with the protocol service calls (e.g., task attribute, task priority, sense data). The attribute is initialized, then may change values as the task ages (e.g. there is no Status at the beginning). Attributes are all represented as [1] instance - [0..1] might also be workable for the attributes that don’t exist when the object is created (or ever).

4. Both the device server and the task manager need to be allowed to invoke Terminate Data Transfer (), not just the device server. As the task manager is processing ABORT TASK and other functions that abort tasks, it is the one that needs to terminate any data transfers in progress. The device server does this when performing the PERSISTENT RESERVE OUT command PREEMPT AND ABORT service action.

Editor’s Note 1: Some change bars are currently duplicated from 07-263

Suggested changes

3.1.4 application client: An object within a SCSI initiator device that is the source of commands and task management function requests.

3.1.18 command: A request describing a unit of work to be performed by a device server.

3.1.31 device server: An object within a logical unit that processes SCSI tasks according to the requirements for task management described in clause 8. See 4.5.20.

3.1.73 operation: A service that may be requested from any object of the class in order to effect behavior. Operations describe what a class is allowed to do and may be a request or a question. A request may change the state of the object but a question should not.

3.1.119 task: An object within the logical unit representing the work associated with a command. See 4.5.~~19~~23.

3.1.120 task priority: The relative scheduling importance of a task having the SIMPLE task attribute among the set of tasks having the SIMPLE task attribute already in the task set. See 8.7.

3.1.121 task tag: ~~An attribute of a task class containing up to 64 bits~~A portion of an I_T_L_Q nexus identifier that uniquely identifies each task for a given I_T_L nexus (see 3.1.49) in a task set (see 3.1.127). See ~~4.5.23.2~~4.8.

3.1.122 task management function: A task manager service capable of being requested by an application client to affect the processing of one or more tasks. See 4.5.24 and clause 7.

3.1.125 task manager: ~~A server~~ An object within a logical unit that controls the sequencing of one or more tasks and processes task management functions. See 4.5.19.

3.1.126 task router: An object in a SCSI target port that routes commands and task management functions between a service delivery subsystem (see 3.1.113) and the appropriate task manager(s) ~~(see 4.5.8)~~. See 4.5.8.

3.1.127 task set: A group of tasks within a logical unit, whose interaction is dependent on the task management (e.g., queuing) and ACA requirements. See 4.5.22.

Editor's Note 2: Reason - adding cross references to attribute sections.

---

4 SCSI architecture model

4.1 Introduction

The purpose of the SCSI architecture model is to:

a) Provide a basis for the coordination of SCSI standards development that allows each standard to be placed into perspective within the overall SCSI architecture model;
b) Establish a layered model in which standards may be developed;
c) Provide a common reference for maintaining consistency among related standards; and
d) Provide the foundation for application compatibility across all SCSI interconnect and SCSI transport protocol environments by specifying generic requirements that apply uniformly to all implementation standards within each functional area.

The development of this standard is assisted by the use of an abstract model. To specify the external behavior of a SCSI system, elements in a system are replaced by functionally equivalent components within this model. Only externally observable behavior is retained as the standard of behavior. The description of internal behavior in this standard is provided only to support the definition of the observable aspects of the model. Those aspects are

limited to the generic properties and characteristics needed for host applications to interoperate with SCSI devices in any SCSI interconnect and SCSI transport protocol environment. The model does not address other requirements that may be essential to some I/O system implementations (e.g., the mapping from SCSI device addresses to network addresses, the procedure for discovering SCSI devices on a network, and the definition of network authentication policies for SCSI initiator devices or SCSI target devices). These considerations are outside the scope of this standard.

The set of SCSI standards specifies the interfaces, functions, and operations necessary to ensure interoperability between conforming SCSI implementations. This standard is a functional description. Conforming implementations may employ any design technique that does not violate interoperability.

The SCSI architecture model is described in terms of classes (see 3.1.13), protocol layers, and service interfaces between classes. As used in this standard, classes are abstractions, encapsulating a set of related functions (i.e., attributes), operations, data types, and other classes. Certain classes are defined by SCSI (e.g., an interconnect), while others are needed to understand the functioning of SCSI but have implementation definitions outside the scope of SCSI (e.g., a task). These classes exhibit well-defined and observable behaviors, but they do not exist as separate physical elements. A class may contain a single attribute ~~(e.g., a task tag)~~ or be a complex entity that contains multiple attributes and performs a set of operations or services on behalf of another class.

Editor's Note 3: Reason: It's not clear that task tag will survive as an attribute, so removing it from the e.g. Many classes have attributes and operations; they're not exclusive.

Service interfaces are defined between distributed classes and protocol layers. The template for a distributed service interface is the client-server model described in 4.2. The structure of a SCSI I/O system is specified in 4.4 by defining the relationship among classes. The set of distributed services to be provided are specified in clause 5 and clause 7.

Requirements that apply to each SCSI transport protocol standard are specified in the SCSI transport protocol service model described in 5.4, 6.4, and 7.12. The model describes required behavior in terms of layers, classes within layers and SCSI transport protocol service transactions between layers.

4.3 The SCSI client-server model

4.3.1 SCSI client-server model overview

As shown in figure 12, each SCSI target device provides services performed by device servers and task management functions performed by task managers. A logical unit is a class that implements one of the device functional models described in the SCSI command standards and processes commands (e.g., reading from or writing to the media). Each pending command defines a unit of work to be performed by the logical unit. Each unit of work is represented within the SCSI target device by a task that may be externally referenced and controlled through requests issued to the task manager.

Figure 12 — SCSI client-server model

All requests originate from application clients residing within a SCSI initiator device. An application client is independent of the interconnect and SCSI transport protocol (e.g., an application client may correspond to the device driver and any other code within the operating system that is capable of managing I/O requests without requiring knowledge of the interconnect or SCSI transport protocol). ~~An application client creates one or more application client tasks each of which issues a command or a task management function. Application client tasks are part of their parent application client. An application client task ceases to exist once the command or task management function ends.~~

Editor's Note 4: Reason: That was about the only use of "application client task", hardly deserving of keeping the term around. Changing "application client task" to mean a task as known on the initiator side (parallel to "task" on the target side).

As described in 4.2, each request takes the form of a procedure call with arguments and a status to be returned. An application client may request processing of a command through a request directed to the device server within a logical unit. Each device service request contains a CDB defining the operation to be performed along with a list of command specific inputs and other parameters specifying how the command is to be processed.

4.5 SCSI classes

4.5.1 SCSI classes overview

Figure 14 shows the main functional classes of the SCSI domain. This standard defines these classes in greater detail.

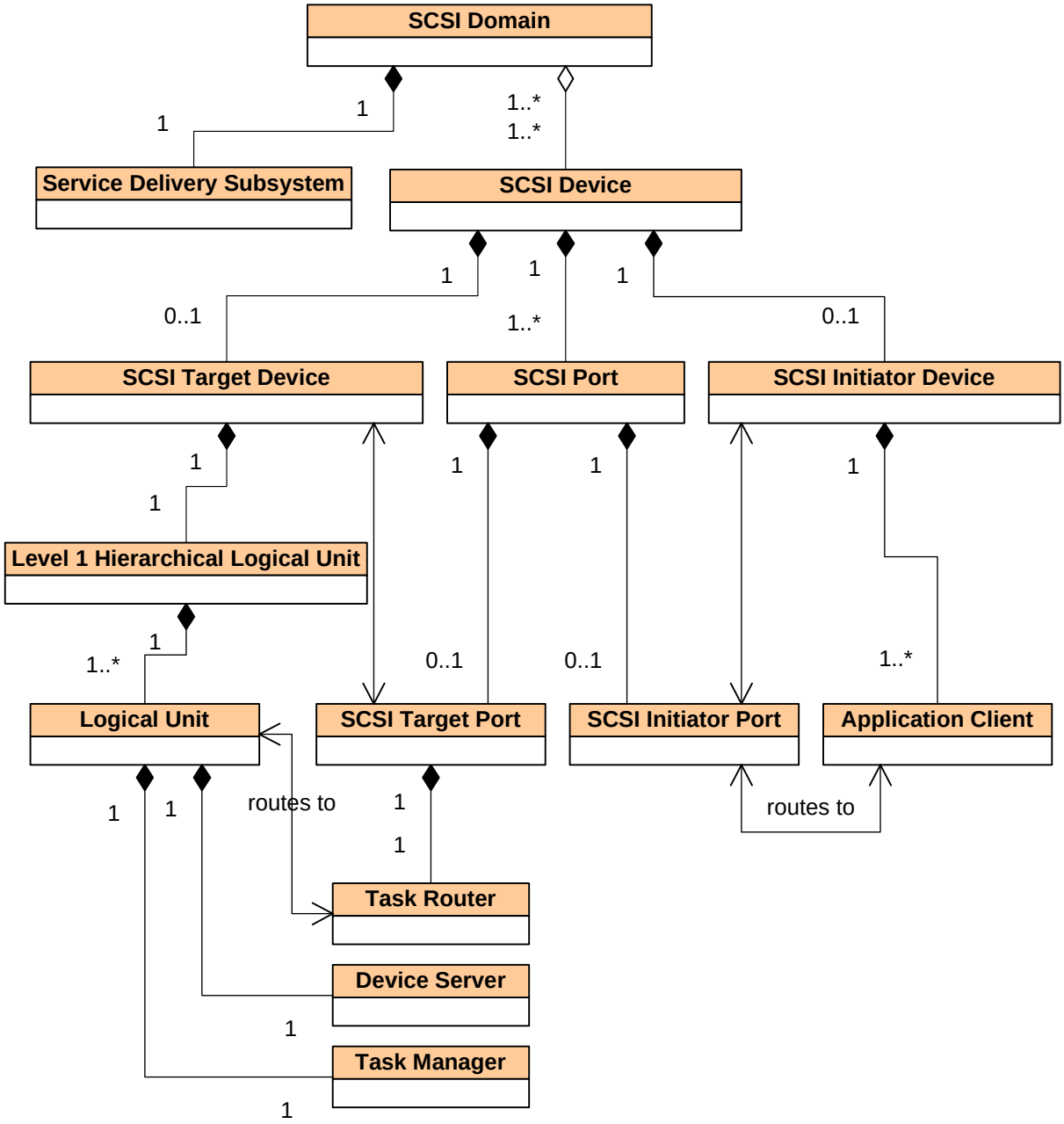

Figure 14 — SCSI Domain class diagram overview [modified - removed AC Task, Task Set, Task]

4.5.2 SCSI Domain class

~~A~~The SCSI Domain class (figure 15) contains the:

a) Service Delivery Subsystem class (see 4.5.3); and
b) SCSI Device class (see 4.5.4) that contains the:
   A) SCSI Port class (see 4.5.5).

Figure 15 — SCSI Domain class diagram

~~Each instance of a SCSI Domain class shall contain the following objects:~~

~~a) one service delivery subsystem;~~
~~b) one or more SCSI devices; and~~
~~c) one or more SCSI ports.~~

Editor's Note 5: Reason: Classes don't have options about what they contain. Objects within the classes do.

Each object in the SCSI Domain class contains:

a) one Service Delivery Subsytem object (see 4.5.3):
b) one or more SCSI Device object (see 4.5.4); and
c) one or more SCSI Port objects (see 4.5.5).

See figure 16 for the instantiation of the minimum set of objects that make up a valid SCSI domain.

Figure 16 — SCSI ~~d~~Domain object diagram

The boundaries of a SCSI domain are established by the system implementor, within the constraints of a specific SCSI transport protocol and associated interconnect standards.

4.5.3 Service Delivery Subsystem class

A Service Delivery Subsystem class connects all the SCSI ports (see 3.1.98) in the SCSI domain, providing a mechanism through which application clients communicate with device servers and task managers (see 4.5.3).

A service delivery subsystem is composed of one or more interconnects that appear to a client or server as a single path for the transfer of requests and responses between SCSI devices.

A service delivery subsystem is assumed to provide error-free transmission of requests and responses between client and server. Although a device driver in a SCSI implementation may perform these transfers through several interactions with its SCSI transport protocol layer, the architecture model portrays each operation, from the viewpoint of the application client, as occurring in one discrete step. The request or response is:

a) considered sent by the sender when the sender passes it to the SCSI port for transmission;
b) in transit until delivered; and
c) considered received by the receiver when it has been forwarded to the receiver via the destination SCSI device's SCSI port.

4.5.4 SCSI Device class

4.5.4.1 SCSI Device class overview

~~See figure 17 for the SCSI Device class diagram.~~

~~A~~The SCSI Device class ~~(see figure 17)~~ contains the:

a) SCSI Port class (see 4.5.5); and
~~b) SCSI Initiator Device class (see 4.5.9), the SCSI Target Device class (see 4.5.14), or both.~~
c) SCSI Initiator Device class (see 4.5.9); and
d) SCSI Target Device class (see 4.5.14).

Figure 17 — SCSI Device class diagram

Editor's Note 6: Reason: Classes don't have options about what they contain. Objects within the classes do.

~~Each instance of a SCSI Device class shall contain:~~

~~a) one or more SCSI ports; and~~
~~b) one SCSI target device, one SCSI initiator device, or both.~~

Each object in the SCSI Device class contains:

a) one or more SCSI Port objects (see 4.5.5); and
b) one SCSI Initiator Device object (see 4.5.9), one SCSI Target Device object (see 4.5.14); or both.

4.5.4.2 SCSI Device Name attribute

A SCSI Device Name attribute contains a name (see 3.1.68) for a SCSI device that is world wide unique within the SCSI transport protocol of each SCSI domain in which the SCSI device has SCSI ports. For each supported SCSI transport protocol, a SCSI device shall have no more than one (i.e., zero or one) SCSI Device Name attribute that is not in the SCSI name string format (see SPC-3). A SCSI device shall have no more than one (i.e., zero or one) SCSI Device Name attribute in the SCSI name string format regardless of the number of SCSI transport protocols supported by the SCSI device. If a SCSI device has a SCSI Device Name attribute in the SCSI name string format then the SCSI device should have only one SCSI Device Name attribute. A SCSI device name shall never change and may be used to persistently identify a SCSI device in contexts where specific references to port names or port identifiers is not required.

A SCSI transport protocol standard may require that a SCSI device include a SCSI Device Name attribute if the SCSI device has SCSI ports in a SCSI domain of that SCSI transport protocol. The SCSI Device Name attribute may be made available to other SCSI devices or SCSI ports in a given SCSI domain in SCSI transport protocol specific ways.

4.5.5 SCSI Port class

4.5.5.1 SCSI Port class overview

AThe SCSI Port class (see figure 18) contains the:

~~a) SCSI Target Port class (see 4.5.6) that contains the:~~
~~A) Task Router class (see 4.5.8);~~
~~b) SCSI Initiator Port class (see 4.5.7.1); or~~
~~c) both.~~
a) SCSI Target Port class (see 4.5.6) that contains the Task Router class (see 4.5.8); and
b) SCSI Initiator Port class (see 4.5.7).

Editor's Note 7: Reason: Classes don't have options about what they contain. Objects within the classes do.

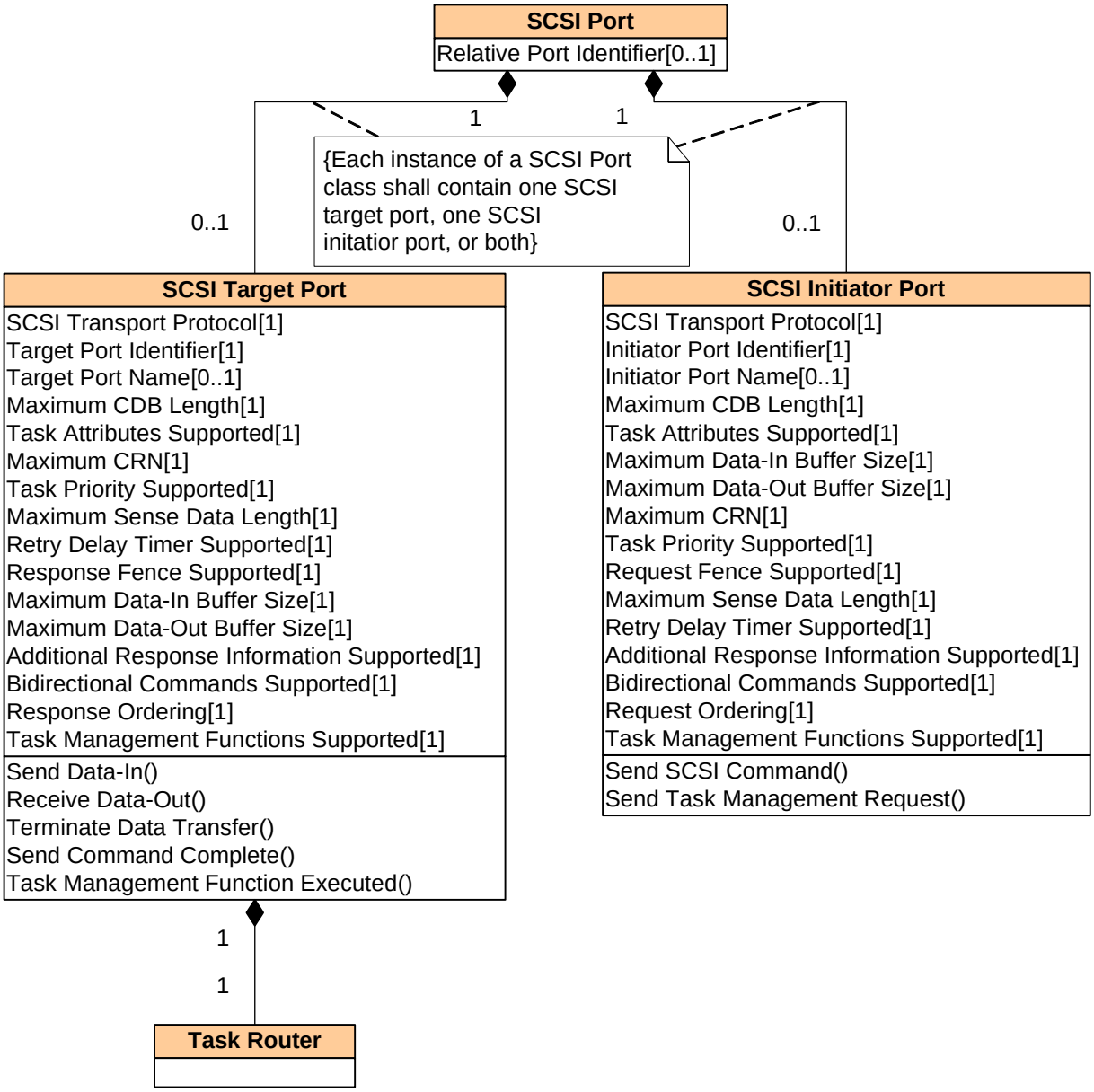

Figure 18 — SCSI Port class diagram [based 07-263 and modified mainly to add operations]

~~Each instance of a SCSI Port class shall contain:~~

~~a) one SCSI target port that shall contain:~~
~~A) one task router;~~
~~b) one SCSI initiator port; or~~
~~c) both.~~

Each object in the SCSI Port class contains:

a) one SCSI Target Port object (see 4.5.6) that contains:
A) one Task Router object (see 4.5.8);
b) one SCSI Initiator Port object (see 4.5.7); or
c) both.

Editor's Note 8: Reason: rewording in terms of objects rather than instances.

4.5.5.2 Relative Port Identifier attribute

The Relative Port Identifier attribute identifies a SCSI target port or a SCSI initiator port relative to other SCSI ports in a SCSI target device and any SCSI initiator devices contained within that SCSI ~~target~~ device. A SCSI target device may assign relative port identifiers to its SCSI target ports and any SCSI initiator ports. If relative port identifiers are assigned, the SCSI target device shall assign each of its SCSI target ports and any SCSI initiator ports a unique relative port identifier from 1 to 65 535. SCSI target ports and SCSI initiator ports share the same number space.

Editor's Note 9: Reason: SCSI initiator devices are not contained within SCSI target devices; they're peer classes under the SCSI Device class.

Relative port identifiers may be retrieved through the Device Identification VPD page (see SPC-3) and the SCSI Ports VPD page (see SPC-3).

The relative port identifiers are not required to be contiguous. The relative port identifier for a SCSI port shall not change once assigned unless physical reconfiguration of the SCSI target device occurs.

4.5.6 SCSI Target Port class

4.5.6.1 SCSI Target Port class overview

~~A~~The SCSI Target Port class (see figure 18 in 4.5.5.1) contains the:

a) Task Router class (see 4.5.8);

Each object in the SCSI Target Port class contains:

a) one Task Router object (see 4.5.8).

The SCSI Target Port class connects SCSI target devices to a service delivery subsystem.

Objects in the SCSI Target Port class invoke the following operations:

a) Device Server class SCSI Command Received () operation (see 4.5.20.2); and
b) Task Manager class Task Management Request Received () operation (see 4.5.21.1).

4.5.6.2 SCSI Transport Protocol attribute

The SCSI Transport Protocol attribute contains the SCSI transport protocol of the SCSI target port. This value may be encoded like the protocol identifier in SPC-4.

4.5.6.3 Targe~~T~~t Port Identifier attribute

The Target Port Identifier attribute contains a target port identifier (see 3.1.117) for a SCSI target port. The target port identifier is a value by which a SCSI target port is referenced within a domain.

4.5.6.4 Target Port Name attribute

A Target Port Name attribute contains an optional name (see 3.1.68) of a SCSI target port that is world wide unique within the SCSI transport protocol of the SCSI domain of that SCSI target port. A SCSI target port may have at most one name. A SCSI target port name shall never change and may be used to persistently identify the SCSI target port.

A SCSI transport protocol standard may require that a SCSI target port include a SCSI target port name if the SCSI target port is in a SCSI domain of that SCSI transport protocol. The SCSI target port name may be made

available to other SCSI devices or SCSI ports in the given SCSI domain in SCSI transport protocol specific ways.

4.5.6.5 Maximum CDB Length attribute

The Maximum CDB Length attribute contains the maximum length of the Execute Command () and Send SCSI Command Received () CDB argument that is supported by the SCSI target port and its SCSI transport protocol.

4.5.6.6 Task Attributes Supported attribute

The Maximum CDB Length attribute contains the Execute Command () and SCSI Command Received () Task Attribute argument values (see table 38 in 8.6.1) that are supported by the SCSI target port and its SCSI transport protocol.

4.5.6.7 Maximum CRN attribute

The Maximum CRN attribute contains the maximum value of the Execute Command () and SCSI Command Received () CRN argument supported by the SCSI target port and its SCSI transport protocol. A value of zero indicates CRN is not supported.

4.5.6.8 Task Priority Supported attribute

The Task Priority Supported attribute indicates if the Execute Command () and Send SCSI Command Received () Task Priority argument is supported by the SCSI target port and its SCSI transport protocol.

4.5.6.9 Maximum Sense Data Length attribute

The Maximum Sense Data Length attribute contains the maximum value of the Execute Command () and Send Command Complete () Sense Data Length argument that is supported by the SCSI target port and its SCSI transport protocol.

4.5.6.10 Retry Delay Timer Supported attribute

The Retry Delay Timer Supported attribute indicates if the Execute Command () and Send Command Complete () Retry Delay Timer argument is supported by the SCSI target port and its SCSI transport protocol.

4.5.6.11 Response Fence Supported attribute

The Response Fence Supported attribute indicates if the Execute Command () and Send Command Complete () Response Fence argument is supported by the SCSI target port and its SCSI transport protocol.

4.5.6.12 Maximum Data-In Buffer Size attribute

The Maximum Data-In Buffer Size attribute contains:

a) the maximum value of the Execute Command () Data-In Buffer Size argument; and
b) the maximum value of the Send Data-In () Application Client Buffer Offset argument plus one (e.g., if the maximum data-in buffer size is 100000000h, then the maximum buffer offset is FFFFFFFFh),

that is supported by the SCSI target port and its SCSI transport protocol.

4.5.6.13 Maximum Data-Out Buffer Size attribute

The Maximum Data-Out Buffer Size attribute contains:

a) the maximum value of the Execute Command () Data-Out Buffer Size argument; and
b) the maximum value of the Receive Data-Out () Application Client Buffer Offset argument plus one (e.g., if the maximum data-in buffer size is 100000000h, then the maximum buffer offset is FFFFFFFFh),

that is supported by the SCSI target port and its SCSI transport protocol.

4.5.6.14 Additional Response Information Supported attribute

The Additional Response Information Supported attribute indicates if the task management procedure call and Task Management Function Executed () Additional Response Information argument is supported by the SCSI target port and its SCSI transport protocol.

4.5.6.15 Bidirectional Commands Supported attribute

The Bidirectional Commands Supported attribute indicates if bidirectional commands are supported by the SCSI target port and its SCSI transport protocol.

4.5.6.16 Response Ordering attribute

The Response Ordering attribute indicates the level of response ordering that the SCSI target port and its SCSI transport protocol ensure. The attribute contains one of the values defined in table 1.

Table 1 — SCSI Target Port class Response Ordering attribute

Value	Description
None	Command and task management function responses may or may not be delivered to the SCSI initiator port in the order in which the device server invoked the Send Command Complete () and Task Management Function Executed () transport protocol service responses.
Command	Command responses are delivered to the SCSI initiator port in the order in which the device server invoked the Send SCSI Command () transport protocol service responses. Task management function responses may or may not be delivered to the SCSI initiator port in the order in which the device server invoked the Task Management Function Executed () transport protocol service responses. There is no ordering between commands and task management function responses.
Full	Commands and task management function responses are delivered to the SCSI initiator port in the order in which the device server invoked the Send Command Complete () and Task Management Function Executed () transport protocol service responses.

4.5.6.17 Task Management Functions Supported attribute

The Task Management Functions Supported attribute indicates the task management procedure calls (see table 34 in 7.1) that are supported by the SCSI target port and its SCSI transport protocol.

4.5.6.18 Send Data-In operation

The Send Data-In operation causes the SCSI target port to perform the Send Data-In transport protocol service request (see 5.4.3.2.1).

4.5.6.19 Receive Data-Out operation

The Receive Data-Out operation causes the SCSI target port to perform the Receive Data-Out transport protocol service request (see 5.4.3.3.1).

4.5.6.20 Terminate Data Transfer operation

The Terminate Data Transfer operation causes the SCSI target port to perform the Terminate Data Transfer transport protocol service request (see 5.4.3.4.1).

4.5.6.21 Send Command Complete operation

The Send Command Complete operation causes the SCSI target port to perform the Send Command Complete transport protocol service response (see 5.4.2.4).

4.5.6.22 Task Management Function Executed operation

The Task Management Function Executed operation causes the SCSI target port to perform the Task Management Function Executed transport protocol service response (see 7.12.4).

Editor's Note 10: Reason: above sections add the protocol services as UML operations

---

4.5.7 SCSI Initiator Port class

4.5.7.1 SCSI Initiator Port class overview

The SCSI Initiator Port class:

a) routes information (e.g., commands and task management functions) between an application client and the services delivery subsystem using the route application client information operation; and
b) connects SCSI initiator devices to a service delivery subsystem.

Objects in the SCSI Initiator Port class invoke the following operations:

a) Application Client class Command Complete Received () operation (see 4.5.10.2); and
b) Application Client class Received Task Management Function Executed () operation (see 4.5.10.3).

4.5.7.2 SCSI Transport Protocol attribute

The SCSI Transport Protocol attribute contains the SCSI transport protocol of the SCSI initiator port. This value may be encoded like the protocol identifier in SPC-4.

4.5.7.3 Initiator Port Identifier attribute

The Initiator Port Identifier attribute contains the initiator port identifier for a SCSI initiator port. The initiator port identifier is a value by which a SCSI initiator port is referenced within a domain.

4.5.7.4 Initiator Port Name attribute

An Initiator Port Name attribute contains an optional name (see 3.1.68) of a SCSI initiator port that is world wideunique within the SCSI transport protocol of the SCSI domain of that SCSI initiator port. A SCSI initiator port may have at most one name. A SCSI initiator port name shall never change and may be used to persistently identify the SCSI initiator port.

A SCSI transport protocol standard may require that a SCSI initiator port include a SCSI initiator port name if the SCSI initiator port is in a SCSI domain of that SCSI transport protocol. The SCSI initiator port name may be made available to other SCSI devices or SCSI ports in the given SCSI domain in SCSI transport protocol specific ways.

4.5.7.5 Maximum CDB Length attribute

The Maximum CDB Length attribute contains the maximum length of the Execute Command () and Send SCSI Command () CDB argument that is supported by the SCSI initiator port and its SCSI transport protocol.

4.5.7.6 Task Attributes Supported attribute

The Maximum CDB Length attribute contains the Execute Command () and Send SCSI Command () Task Attribute argument values (see table 38 in 8.6.1) that are supported by the SCSI initiator port and its SCSI transport protocol.

4.5.7.7 Maximum Data-In Buffer Size attribute

The Maximum Data-In Buffer Size attribute contains the maximum value of the Execute Command () and Send SCSI Command () Data-In Buffer Size argument that is supported by the SCSI initiator port and its SCSI transport protocol.

4.5.7.8 Maximum Data-Out Buffer Size attribute

The Maximum Data-Out Buffer Size attribute contains the maximum value of the Execute Command () and Send SCSI Command () Data-Out Buffer Size argument that is supported by the SCSI initiator port and its SCSI transport protocol.

4.5.7.9 Maximum CRN attribute

The Maximum CRN attribute contains the maximum value of the Execute Command () and Send SCSI Command () CRN argument supported by the SCSI initiator port and its SCSI transport protocol. A value of zero indicates CRN is not supported .

4.5.7.10 Task Priority Supported attribute

The Task Priority Supported attribute indicates if the Execute Command () and Send SCSI Command () Task Priority argument is supported by the SCSI initiator port and its SCSI transport protocol.

4.5.7.11 Request Fence Supported attribute

The Request Fence Supported attribute indicates if the Execute Command () and Send SCSI Command () Request Fence argument is supported by the SCSI initiator port and its SCSI transport protocol.

4.5.7.12 Maximum Sense Data Length attribute

The Maximum Sense Data Length attribute contains the maximum value of the Execute Command () and Command Complete Received () Sense Data Length argument that is supported by the SCSI initiator port and its SCSI transport protocol.

4.5.7.13 Retry Delay Timer Supported attribute

The Retry Delay Timer Supported attribute indicates if the Execute Command () and Command Complete Received () Retry Delay Timer argument is supported by the SCSI initiator port and its SCSI transport protocol.

4.5.7.14 Additional Response Information Supported attribute

The Additional Response Information Supported attribute indicates if the task management procedure call and Received Task Management Function Executed () Additional Response Information argument is supported by the SCSI initiator port and its SCSI transport protocol.

4.5.7.15 Bidirectional Commands Supported attribute

The Bidirectional Commands Supported attribute indicates if bidirectional commands are supported by the SCSI initiator port and its SCSI transport protocol.

4.5.7.16 Request Ordering attribute

The Request Ordering attribute indicates the level of request ordering that the SCSI initiator port and its SCSI transport protocol ensure. The attribute contains one of the values defined in table 2.

Table 2 — SCSI Initiator Port class Request Ordering attribute

Value	Description
None	Commands and task management functions may or may not be delivered to the SCSI target port in the order in which the application client invoked the Execute Command () procedure call and the task management procedure calls (i.e., the Send SCSI Command () and Send Task Management Request () transport protocol service requests).
Command	Commands are delivered to the SCSI target port in the order in which the application client invoked the Execute Command () procedure calls (i.e., the Send SCSI Command () transport protocol service requests). Task management functions may or may not be delivered to the SCSI target port in the order in which the application client invoked the task management procedure calls (i.e., Send Task Management Request () transport protocol service requests). There is no ordering between commands and task management functions.
Full	Commands and task management functions are delivered to the SCSI target port in the order in which the application client invoked the Execute Command () procedure call and the task management procedure calls (i.e., the Send SCSI Command () and Send Task Management Request () transport protocol service requests).

4.5.7.17 Task Management Functions Supported attribute

The Task Management Functions Supported attribute indicates the task management procedure calls (see table 34 in 7.1) that are supported by the SCSI initiator port and its SCSI transport protocol.

4.5.7.18 Send SCSI Command operation

The Send SCSI Command operation causes the SCSI initiator port to perform the Send SCSI Command transport protocol service request (see 5.4.2.2).

4.5.7.19 Send Task Management Request operation

The Send Task Management Request operation causes the SCSI initiator port to perform the Send Task Management Request transport protocol service request (see 7.12.2).

Editor's Note 11: Reason: above sections add the protocol services as UML operations

---

4.5.8 Task Router class

The Task Router class routes information (e.g., commands and task management functions) between a logical unit and a service delivery subsystem using the route task operation.

The task router routes commands and task management functions as follows:

a) Commands addressed to a valid logical unit are routed to the task manager in the specified logical unit;
b) Commands addressed to an incorrect logical unit are handled as described in 5.8.4;
c) Task management functions with I_T_L nexus scope (e.g., ABORT TASK SET, CLEAR TASK SET, CLEAR ACA, LOGICAL UNIT RESET, QUERY TASK SET, and QUERY UNIT ATTENTION) or I_T_L_Q nexus scope (e.g., ABORT TASK and QUERY TASK) addressed to a valid logical unit are routed to the task manager in the specified logical unit;

d) Task management functions with an I_T nexus scope (e.g., I_T NEXUS RESET) are routed to the task manager in each logical unit about which the task router knows; and
e) Task management functions with I_T_L nexus scope or I_T_L_Q nexus scope addressed to an incorrect logical unit are handled as described in 7.12.

In some transport protocols, the task router may check for overlapped task tags on commands (see 5.8.3).

4.5.9 SCSI Initiator Device class

~~A SCSI Initiator Device class (see figure 19) is a SCSI Device class that contains the:~~

~~a) Application Client class (see 4.5.10) that contains the:~~
~~A) Application Client Task class (see 4.5.11).~~

The SCSI Initiator Device class (see figure 19) contains the:

a) Application Client class (see 4.5.10) that contains the:
A) Application Client Task Set class (see 4.5.11) that contains the Application Client Task class (see 4.5.13); and
B) Application Client Task Management Function class (see 4.5.14).

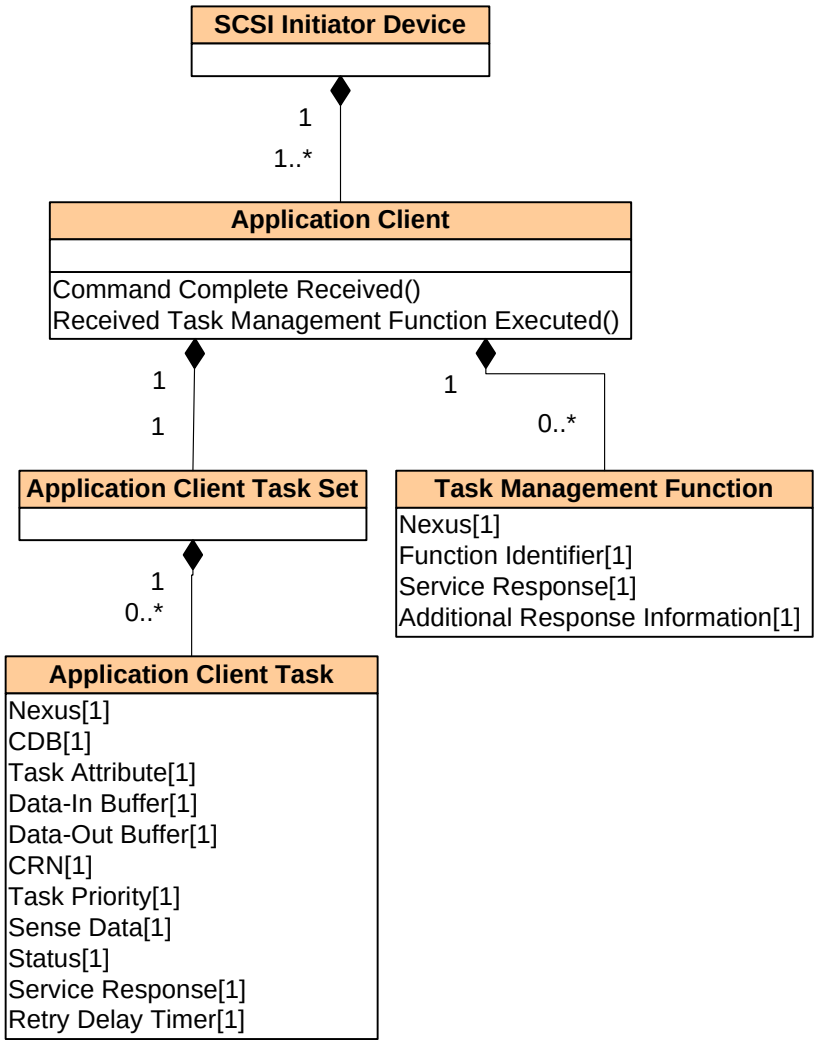

Figure 19 — SCSI Initiator Device class diagram [modified]

~~Each instance of a SCSI Initiator Device class shall contain the following objects:~~

~~a) one or more application clients that contain:~~
~~A) zero or more application client tasks.~~

Each object in the SCSI Initiator Device class contains:

a) one or more Application Client objects (see 4.5.10) that contain:
A) one Application Client Task Set object (see 4.5.11) that contains zero or more Application Client Tasks objects (see 4.5.13); and
B) zero or more Application Client Task Management Function objects (see 4.5.14).

4.5.10 Application Client class

4.5.10.1 Application Client class overview

~~An Application Client class contains zero or more application client tasks.:~~

The Application Client class (see figure 19 in 4.5.9) contains the:

a) Application Client Task Set class (see 4.5.11) containing the Application Client Task class (see 4.5.13); and
b) Task Management Function class (see 4.5.14).

Editor's Note 12: Reason: make each class section list other classes it contains. Then list the object rules (which are constrained with numbers like one, zero or more, etc.)

Each object in the Application Client class contains:

a) one Application Client Task Set object (see 4.5.11) containing zero or more Application Client Task objects (see 4.5.13); and
b) zero or more Task Management Function objects (see 4.5.14).

~~An Application Client class originates commands by issuing Send SCSI Command requests (see 5.4.2)~~

~~An Application Client class originates task management requests by issuing a Function name service request (see clause 7).~~

Objects in the Application Client class invoke the following operations:

a) SCSI Initiator Port class Send SCSI Command () operation (see 4.5.7.18); and
b) SCSI Initiator Port class Send Task Management Request () operation (see 4.5.7.19).

An application client may request processing of a task management function through a request directed to the task manager within the logical unit. The interactions between the task manager and application client when a task management request is processed are shown in 7.13.

4.5.10.2 Command Complete Received operation

The Command Complete Received operation causes the application client to perform the Command Complete Received transport protocol service confirmation (see 5.4.2.5).

4.5.10.3 Received Task Management Function Executed operation

The Received Task Management Function Executed operation causes the application client to perform the Received Task Management Function Executed transport protocol service confirmation (see 7.12.5).

Editor's Note 13: Reason: above sections add the protocol services as UML operations

---

4.5.11 Application Client Task Set class

The Application Client Task Set class (see figure 19 in 4.5.9) contains the:

a) Application Client Task class (see 4.5.13).

Each object in the Application Client Task Set class contains:

a) zero or more Application Client Task objects (see 4.5.13).

~~**4.5.12 Application Client Task class**~~

~~An Application Client Task class (see figure 19) shall be substituted with:~~

~~a) a Task class (see 4.5.12); or~~
~~b) a Task Management Function class (see 4.5.13).~~

~~An Application Client Task class is the source for a single command or a single task management function.~~

Editor's Note 14: Reason: Application client task old definition not really necessary. A new definition paralleling the Task Set on the target side is more useful.

4.5.13 Application Client Task class

4.5.13.1 Application Client Task class overview

An Application Client Task class ~~is an Application Client class that~~ represents a command (see clause 5).

Each instance of an Application Client Task class represents the work associated with a command. A new command causes the creation of an application client task. The application client task persists until a task complete response is sent or until the task is ended by a task management function or exception condition. For an example of the processing for a command see 5.7.

4.5.13.2 Nexus attribute

The Nexus attribute contains:

a) an initiator port identifier;
b) a target port identifier;
c) a LUN; and
d) a task tag.

The Nexus attribute uniquely identifies an application client task within the application client task set.

Editor's Note 15: It might be better to just include those values as attributes - Initiator Port Identifier, Target Port Identifier, LUN, and Task Tag - instead of combining them into a Nexus attribute.

The SCSI Initiator Port class Send SCSI Command () operation uses this attribute.

4.5.13.3 CDB ~~A~~attribute

The CDB attribute contains a CDB (see 5.2 and SPC-3).

The SCSI Initiator Port class Send SCSI Command () operation uses this attribute.

4.5.13.4 Task Attribute attribute

The Task Attribute attribute contains a task attribute (see 8.6).

The SCSI Initiator Port class Send SCSI Command () operation uses this attribute.

4.5.13.5 Data-In Buffer attribute

The Data-In Buffer attribute represents the application client data-in buffer, if any (see 5.4.3.1), and its size.

The SCSI Initiator Port class Send SCSI Command () operation uses this attribute.

4.5.13.6 Data-Out Buffer attribute

The Data-Out Buffer attribute represents the application client data-out buffer, if any (see 5.4.3.1), and its size.

The SCSI Initiator Port class Send SCSI Command () operation uses this attribute.

4.5.13.7 CRN attribute

The CRN attribute contains the CRN, if any (see 5.1).

The SCSI Initiator Port class Send SCSI Command () operation uses this attribute.

4.5.13.8 Task Priority attribute

The Task Priority attribute contains the task priority, if any (see 8.7).

The SCSI Initiator Port class Send SCSI Command () operation uses this attribute.

4.5.13.9 Sense Data attribute

The Sense Data attribute represents the sense data buffer (see 5.1 and 5.8.6) and its size.

The application client shall initialize this attribute to zero. The Application Client class Command Complete Received () operation updates this attribute.

4.5.13.10 Status attribute

The Status attribute contains the command completion status (see 5.1 and 5.3).

The application client shall initialize this attribute to zero. The Application Client class Command Complete Received () operation updates this attribute.

4.5.13.11 Service Response attribute

The Service Response attribute contains the service response (see 5.1).

The application client shall initialize this attribute to zero. The Application Client class Command Complete Received () operation updates this attribute.

4.5.13.12 Retry Delay Timer attribute

The Retry Delay Timer attribute contains the retry delay timer, if any (see 5.1).

The application client shall initialize this attribute to zero. The Application Client class Command Complete Received () operation updates this attribute.

Editor's Note 16: Reason: The initiator version of Task has most of the same attributes as the target side, but they're not exactly the same.

4.5.14 Application Client Task Management Function class

4.5.14.1 Application Client Task Management Function overview

An Application Client Task Management Function class ~~is an Application Client class that~~ represents a SCSI task management function (see clause 7).

~~**4.5.14.2 Task Management Request_attribute**~~

~~The Task Management Request_attribute contains a task management request (see clause 7).~~

4.5.14.3 Function Identifier attribute

The Function Identifier attribute contains the function identifier (see clause 7).

4.5.14.4 Nexus attribute

The Nexus attribute contains:

a) an initiator port identifier;
a) a target port identifier;
b) a LUN, if the task management function has a scope of I_T_L nexus or I_T_L_Q nexus; and
c) a task tag, if the task management function has a scope of I_T_L_Q nexus.

Editor's Note 17: It might be better to just include those values as attributes - Initiator Port Identifier, Target Port Identifier, LUN, and Task Tag - instead of combining them into a Nexus attribute.

4.5.14.5 Service Response attribute

The Service Response attribute contains the service response (see 5.1).

The application client shall initialize this attribute to zero. The Application Client class Command Complete Received () operation updates this attribute

4.5.14.6 Additional Response Information attribute

The Additional Response Information attribute contains the additional response information, if any (see 7.1).

The application client shall initialize this attribute to zero. The Application Client class Received Task Management Function Executed () operation updates this attribute.

Editor's Note 18: Reason: The initiator version of Task has most of the same attributes as the target side, but they're not exactly the same.

4.5.15 Level 1 Hierarchical Logical Unit class

~~A~~The Level 1 Hierarchical Logical Unit class (see figure 21) contains the:

a) Logical Unit class (see 4.5.19);
a) Well Known Logical Unit class (see 4.5.24); and
a) Level 2 Hierarchical Logical Unit class (see 4.5.16) that contains the:
   A) Logical Unit class;
   B) Well Known Logical Unit class; and
   C) Level 3 Hierarchical Logical Unit class (see 4.5.17) that contains the:
      a) Logical Unit class;
      b) Well Known Logical Unit class; and
      c) Level 4 Hierarchical Logical Unit class (see 4.5.18) that contains the:
         A) Logical Unit class; and
         B) Well Known Logical Unit class.

Figure 20 — Level 1 Hierarchical Logical Unit class

~~Each instance of a Level 1 Hierarchical Logical Unit class shall contain the following objects:~~

~~a) at least one logical unit or well known logical unit;~~

~~b) zero or more logical units;~~
~~a) zero or more well known logical units;~~
~~b) zero or more level 2 hierarchical logical units;~~
~~c) zero or more level 3 hierarchical logical units; or~~
~~d) zero or more level 4 hierarchical logical units.~~

Each object in the Level 1 Hierarchical Logical Unit class contains:

a) at least one Logical Unit object (4.5.19) or at least one Well Known Logical Unit object (see 4.5.24); and
b) zero or more Level 2 Hierarchical Logical Unit objects that contain;
   A) zero or more Level 3 Hierarchical Logical Unit objects that contain zero or more Level 4 Hierarchical Logical Unit objects.

Logical units and well known logical units at each level in the hierarchical logical unit structure are referenced by one of the following address methods:

a) Peripheral device address method (see 4.6.7);
b) Flat space addressing method (see 4.6.8);
c) Logical unit address method (see 4.6.9); or
d) Extended logical unit addressing method (see 4.6.10).

All peripheral device addresses, except LUN 0 (see 4.6.4), default to vendor specific values. All addressable entities, except well known logical units (see 4.5.24), may default to vendor specific values or may be defined by an application client (e.g., by the use of SCC-2 configuration commands).

Within the hierarchical logical unit structure there may be SCSI devices each of which contain a SCSI target device that:

a) has multiple logical units that are accessible through target ports in one SCSI domain; and
b) transfer SCSI operations to a SCSI target device in another SCSI domain through a SCSI initiator device and it's associated SCSI initiator ports.

When using the peripheral device addressing method or the logical unit address method the SCSI domains accessed by these SCSI initiator ports are referred to as buses. A SCSI target device that has SCSI devices attached to these buses shall assign numbers, other than zero, to those buses. The bus numbers shall be used as components of the logical unit numbers to the logical units attached to those buses, as described in 4.6.7 and 4.6.9.

When using the peripheral device addressing method or the logical unit address method SCSI devices shall assign a bus number of zero to all the logical units within the SCSI target device that are not connected to another SCSI domain.

4.5.16 Level 2 Hierarchical Logical Unit class

~~A Level 2 Hierarchical Logical Unit class is a Hierarchical Logical Unit class placed at level 2 within the hierarchical logical unit structure.~~

The Level 2 Hierarchical Logical Unit class (see figure 21 in 4.5.15) contains the:

a) Logical Unit class (4.5.19) class;
b) Well Known Logical Unit class (see 4.5.24); and
c) Level 3 Hierarchical Logical Unit class.

Each object in the Level 2 Hierarchical Logical Unit class contains:

a) zero or one Logical Unit objects (4.5.19) or zero or one Well Known Logical Unit objects (see 4.5.24); and
b) zero or more Level 3 Hierarchical Logical Unit objects.

Editor's Note 19: Reason: Making all these class descriptions consistent

---

All logical units and well known logical units contained within level 2 hierarchical logical unit shall have a Dependent Logical Unit attribute (see 4.5.19.4).

4.5.17 Level 3 Hierarchical Logical Unit class

~~A Level 3 Hierarchical Logical Unit class is a Hierarchical Logical Unit class placed at level 3 within the hierarchical logical unit structure.~~

The Level 3 Hierarchical Logical Unit class (see figure 21 in 4.5.15) contains the:

a) Logical Unit class (4.5.19) class;
b) Well Known Logical Unit class (see 4.5.24); and
c) Level 4 Hierarchical Logical Unit class.

Each object in the Level 3 Hierarchical Logical Unit class contains:

a) zero or one Logical Unit objects (4.5.19) or zero or one Well Known Logical Unit objects (see 4.5.24); and
b) zero or more Level 4 Hierarchical Logical Unit objects.

All logical units and well known logical units contained within level 3 hierarchical logical unit shall have a Dependent Logical Unit attribute (see 4.5.19.4).

4.5.18 Level 4 Hierarchical Logical Unit class

~~A Level 4 Hierarchical Logical Unit class is a Hierarchical Logical Unit class placed at level 4 within the hierarchical logical unit structure.~~

The Level 4 Hierarchical Logical Unit class (see figure 21 in 4.5.15) contains the:

a) Logical Unit class (4.5.19) class; and
b) Well Known Logical Unit class (see 4.5.24).

Each object in the Level 4 Hierarchical Logical Unit class contains:

a) zero or one Logical Unit objects (4.5.19) or zero or one Well Known Logical Unit objects (see 4.5.24).

All logical units and well known logical units contained within level 4 hierarchical logical unit shall have a Dependent Logical Unit attribute (see 4.5.19.4).

4.5.19 Logical Unit class

4.5.19.1 Logical Unit class overview

~~A~~The Logical Unit class (see figure 22) contains the:

a) Device Server class (see 4.5.20);
b) Task Manager class (see 4.5.21); and
c) Task Set class (see 4.5.22); and
d) Task Management Function class (see 4.5.24).

~~A~~The Logical Unit class (see figure 22) may be substituted with the:

a) Well Known Logical Unit class (see 4.5.19.1); or
b) Level 1 Hierarchical Logical Unit class, Level 2 Hierarchical Logical Unit class, Level 3 Hierarchical Logical Unit class, or Level 4 Hierarchical Logical Unit class.

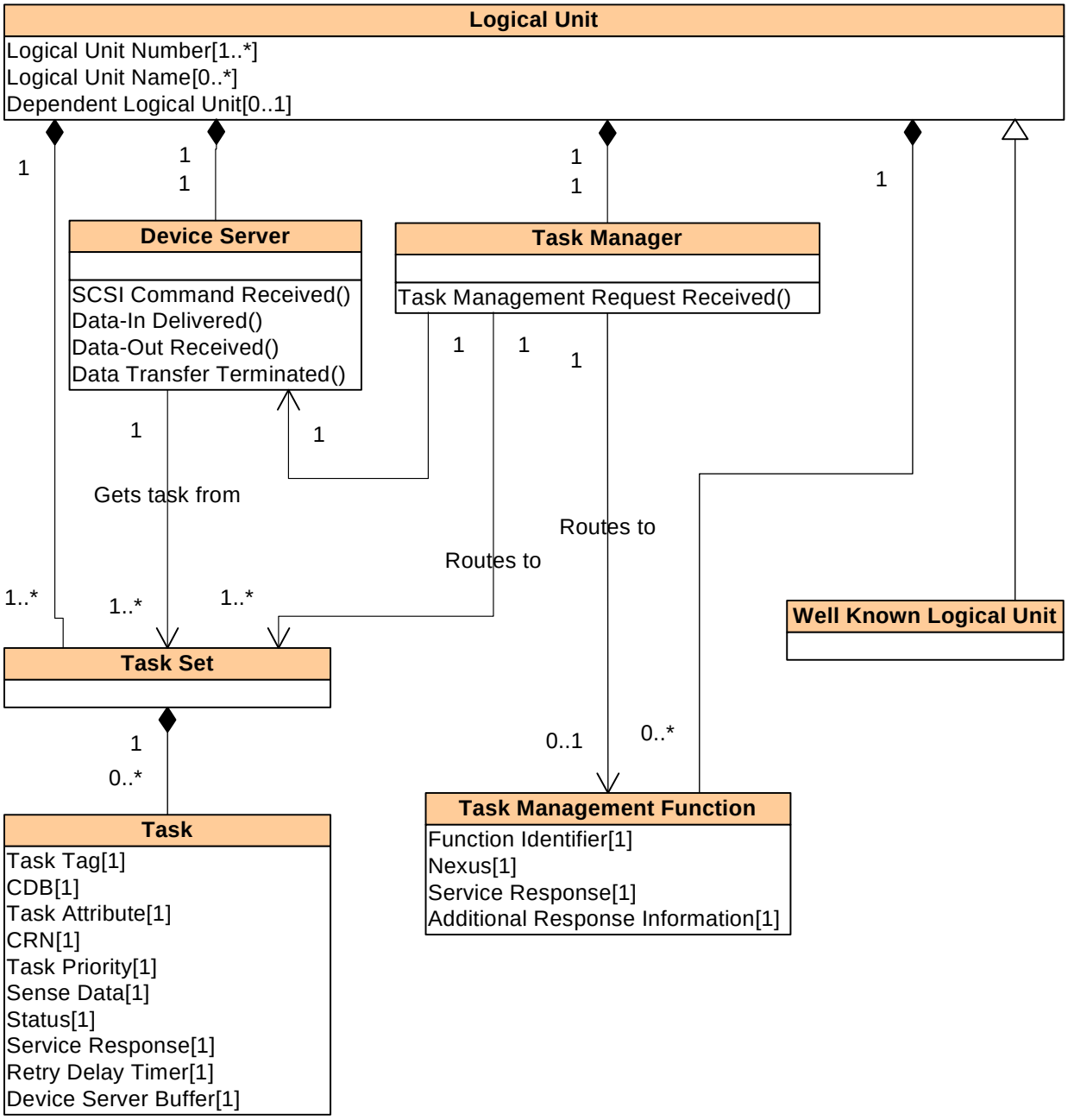

Figure 22 — Logical Unit class diagram [modified]

~~Each instance of a Logical Unit class shall contain the following objects:~~

- ~~a) one device server;~~
- ~~b) one task manager; and~~
- ~~c) one or more task sets.~~

Each object in the Logical Unit class contains:

- a) one Device Server object (see 4.5.20);
- b) one Task Manager object (see 4.5.21); and
- c) one or more Task Set objects (see 4.5.22); and
- d) zero or more Task Management Function objects (see 4.5.24).

A logical unit is the class to which commands are sent. One of the logical units within the SCSI target device shall be accessed using the logical unit number zero or the REPORT LUNS well-known logical unit number.

If the logical unit inventory changes for any reason (e.g., completion of initialization, removal of a logical unit, or creation of a logical unit), then the device server shall establish a unit attention condition (see 5.8.7) for the initiator port associated with every I_T nexus, with the additional sense code set to REPORTED LUNS DATA HAS CHANGED.

4.5.19.2 Logical Unit Number attribute

A Logical Unit Number attribute identifies the logical unit within a SCSI target device when accessed ~~by~~through a SCSI target port. If any logical unit within the scope of a SCSI target device includes one or more dependent logical units (see 4.5.19.4) in its composition, then all logical unit numbers within the scope of the SCSI target device shall have the format described in 4.6.6. If there are no dependent logical units within the scope of the SCSI target device, the logical unit numbers should have the format described in 4.6.5.

The 64-bit quantity called a LUN is the Logical Unit Number attribute defined by this standard. The fields containing the acronym LUN that compose the Logical Unit Number attribute are historical nomenclature anomalies, not Logical Unit Number attributes. LogicaL Unit Number attributes having different values represent different logical units, regardless of any implications to the contrary in 4.6 (e.g., LUN 00000000 00000000h is a different logical unit from LUN 40000000 00000000h and LUN 00FF0000 00000000h is a different logical unit from LUN 40FF0000 00000000h).

Logical unit number(s) are required as follows:

a) If access controls (see SPC-3) are not in effect, one logical unit number per logical unit; or
b) If access controls are in effect, one logical unit number per SCSI initiator port that has access rights plus one default logical unit number per logical unit.

See 4.6 for a definition of the construction of logical unit numbers to be used by SCSI target devices. Application clients should use only those logical unit numbers returned by a REPORT LUNS command. The task router shall respond to logical unit numbers other than those returned by a REPORT LUNS command (i.e., incorrect logical unit numbers) as specified in 5.8.4 and 7.12.

4.5.19.3 Logical Unit Name attribute

A Logical Unit Name attribute identifies a name (see 3.1.68) for a logical unit that is not a well known logical unit.

A logical unit name shall be world wide unique. A logical unit name shall never change and may be used to persistently identify a logical unit.

Logical unit name(s) are required as follows:

a) one or more logical unit names if the logical unit is not a well-known logical unit; or
b) zero logical unit names in the logical unit is a well-known logical unit.

4.5.19.4 Dependent Logical Unit attribute

A Dependent Logical Unit attribute identifies a logical unit that is addressed via a hierarchical logical unit that resides at a lower numbered level in the hierarchy (i.e., no logical unit within level 1 contains a Dependent Logical Unit attribute while all logical units within level 2, level 3, and level 4 do contain a Dependent Logical Unit attribute).

Any instance of a Logical Unit class that contains Dependent Logical Unit attribute shall utilize the hierarchical logical unit number structure defined in 4.6.6. If any logical unit within a SCSI target device includes Dependent Logical Unit attribute, then:

a) all logical units within the SCSI target device shall format all logical unit numbers as described in 4.6.6; and
b) logical unit number zero or the REPORT LUNS well-known logical unit (see SPC-3) shall set the HISUP bit to one in the standard INQUIRY data.

4.5.20 Device Server class

4.5.20.1 Device Server class overview

The Device Server class processes commands.

Objects in the Device Server class invoke the following operations while processing commands:

a) SCSI Target Port class Send Data-In () operation (see 4.5.6.18);
b) SCSI Target Port class Receive Data-Out () operation (see 4.5.6.19);
c) SCSI Target Port class Terminate Data Transfer () operation (see 4.5.6.20); and
d) SCSI Target Port class Send Command Complete () operation (see 4.5.6.21).

4.5.20.2 SCSI Command Received operation

The SCSI Command Received operation causes the device server to perform the SCSI Command Received transport protocol service indication (see 5.4.2.3).

4.5.20.3 Data-In Delivered operation

The Data-In Delivered operation causes the device server to perform the Data-In Delivered transport protocol service confirmation (see 5.4.3.2.2).

4.5.20.4 Data-Out Received operation

The Data-Out Received operation causes the device server to perform the Data-Out Received transport protocol service confirmation (see 5.4.3.3.2).

4.5.20.5 Data Transfer Terminated operation

The Data Transfer Terminated operation causes the device server to perform the Data Transfer Terminated transport protocol service confirmation (see 5.4.3.4.3).

Editor's Note 20: Reason: Adding protocol services as UML operations

---

4.5.21 Task Manager class

The Task Manager class:

a) receive tasks from a task router;
b) place tasks into a task set;
c) ~~a)~~controls the sequencing of one or more tasks within a logical unit; and
d) ~~b)~~processes the task management functions (see clause 7).

Objects in the Task Manager class invoke the following operations while processing task management functions:

a) SCSI Target Port class Terminate Data Transfer () operation (see 4.5.6.20); and
b) SCSI Target Port class Task Management Function Executed () operation (see 4.5.6.22).

4.5.21.1 Task Management Request Received operation

The Task Management Request Received operation causes the device server to perform the Task Management Request Received transport protocol service indication (see 7.12.3).

Editor's Note 21: Reason: Adding protocol services as UML operations

---

4.5.22 Task Set class

~~A~~The Task Set class (see figure 22 in 4.5.19) contains ~~a~~the Task class (see 4.5.23).

~~Each instance of a Task Set class shall contain the following objects:~~

~~a) zero or more tasks.~~

Each object in the Task Set class contains:

a) zero or more Task objects (see 4.5.23);

The interactions among the tasks in a task set are determined by the requirements for task set management specified in clause 8 and the ACA requirements specified in 5.8.1. The number of task sets per logical unit and the boundaries between task sets are governed by the TST field in the Control mode page (see SPC-3).

4.5.23 Task class

4.5.23.1 Task class overview

A Task class represents the work associated with a command. ~~There shall be one Task class for each task that the device server has not started processing.~~

~~A task is represented by an I_T_L_Q nexus (see 4.7) and is composed of:~~

~~a) A definition of the work to be performed by the logical unit in the form of a command;~~
~~b) A Task attribute (see 8.6) that allows the application client to specify processing relationships between various tasks in the task set; and~~
~~c) Optionally, a task priority (see 8.7).~~

Editor's Note 22: Reason: Task Priority, CDB, etc. are now defined as attributes.

~~**4.5.23.2 Task Tag attribute**~~

~~A Task Tag attribute identifies a command. The I_T_L_Q nexus representing a task includes a task tag, allowing many uniquely identified tagged tasks to be present in a single task set. A task tag is composed of up to 64 bits.~~

~~A SCSI initiator device assigns task tag values for each I_T_L_Q nexus in a way that ensures that the nexus uniqueness requirements stated in this subclause are met. Transport protocols may define additional restrictions on task tag assignment (e.g., restricting task tag length, requiring task tags to be unique per I_T nexus or per I_T_L nexus, or sharing task tag values with other uses such as task management functions).~~

~~An I_T_L_Q nexus that is in use (i.e., during the interval bounded by the events specified in 5.5) shall be unique as seen by the SCSI initiator port originating the command and the logical unit to which the command was addressed, otherwise an overlapped command condition exists (see 5.8.3). An I_T_L_Q nexus is unique if one or more of its components is unique within the specified time interval.~~

~~A SCSI initiator device shall not create more than one task from a specific SCSI initiator port having identical values for the target port identifier, logical unit number, and task tag.~~

Editor's Note 23: Reason: Moving above text (for now). It has more initiator rules than target rules, so doesn't belong in the target class section. Also, Task Tag is part of Nexus.

4.5.23.3 Nexus attribute

The Nexus attribute contains:

a) an initiator port identifier;
b) a relative target port identifier; and

c) a task tag.

The Nexus attribute uniquely identifies a task within the task set.

Editor's Note 24: Reason: Since a Task object is stored in a task set, the LUN is already known and is not really part of the Nexus attribute.

4.5.23.4 CDB attribute

The CDB attribute contains a CDB (see 5.2 and SPC-3).

The Device Server class SCSI Command Received () operation updates this attribute.

4.5.23.5 Task Attribute attribute

The Task Attribute attribute contains a task attribute (see 8.6).

The Device Server class SCSI Command Received () operation updates this attribute.

4.5.23.6 CRN attribute

The CRN attribute contains the CRN, if any (see 5.1).

The Device Server class SCSI Command Received () operation updates this attribute.

4.5.23.7 Task Priority attribute

The Task Priority attribute contains the task priority, if any (see 8.7).

The Device Server class SCSI Command Received () operation updates this attribute.

4.5.23.8 Sense Data attribute

The Sense Data attribute represents the sense data buffer (see 5.1 and 5.8.6), if any, and its size.

The SCSI Target Port class Send Command Complete () operation uses this attribute.

4.5.23.9 Status attribute

The Status attribute contains the command completion status (see 5.1 and 5.3).

The SCSI Target Port class Send Command Complete () operation uses this attribute.

4.5.23.10 Service Response attribute

The Service Response attribute contains the service response (see 5.1).

The SCSI Target Port class Send Command Complete () operation uses this attribute.

4.5.23.11 Retry Delay Timer attribute

The Retry Delay Timer attribute contains the retry delay timer, if any (see 5.1).

The SCSI Target Port class Send Command Complete () operation uses this attribute.

4.5.23.12 Device Server Buffer attribute

The Device Server Buffer attribute represents the device server buffer, if any (see 5.1), and its size.

The SCSI Target Port class Send Data-In () operation and Receive Data-Out () operation use this attribute.

4.5.24 Task Management Function class

4.5.24.1 Task Management Function overview

The Task Management Function class represents a SCSI task management function (see clause 7).

4.5.24.2 Function Identifier attribute

The Function Identifier attribute contains the function identifier (see clause 7).

4.5.24.3 Nexus attribute

The Nexus attribute identifies the nexus affected by the task management function (see clause 7).

If the Nexus is an I_T_L_Q nexus, this attribute contains the task tag of the affected task.

4.5.24.4 Service Response attribute

The Service Response attribute contains the service response (see 5.1).

The application client shall initialize this attribute to zero. The Device Server class Command Complete Received () operation updates this attribute

4.5.24.5 Additional Response Information attribute

The Additional Response Information attribute contains the additional response information, if any (see 7.1).

The application client shall initialize this attribute to zero. The Device Server class Received Task Management Function Executed () operation updates this attribute.

4.5.25 Well Known Logical Unit class

~~A~~The Well Known Logical Unit class is a Logical Unit class (see 4.5.19.1) with the additional characteristics defined in this subclause.

Well known logical units are addressed using the well known logical unit addressing method (see 4.6.11) of extended logical unit addressing (see 4.6.10). Each well known logical unit has a well known logical unit number (W-LUN). W-LUN values are defined in SPC-3.

If a SCSI target port receives a W-LUN and the well known logical unit specified by the W-LUN does not exist, the task router shall follow the rules for selection of incorrect logical units described in 5.8.4 and 7.12.

If a well known logical unit is supported within a SCSI target device, then that logical unit shall support all the commands defined for it.

Access to well known logical units shall not be affected by access controls.

All well known logical units:

a) Shall not have logical unit names; and
b) Shall identify themselves using the SCSI target device names of the SCSI device in which they are contained.

NOTE 1 - A SCSI target device may have multiple SCSI target device names if the SCSI target device supports multiple SCSI transport protocols (see 4.5.14).

The name of the well known logical unit may be determined by issuing an INQUIRY command requesting the Device Identification VPD page (see SPC-3).

~~4.7 The nexus object~~Nexus

~~The nexus object represents a relationship between~~ A nexus identifies a SCSI initiator port, a SCSI target port, optionally a logical unit, and optionally a task.

~~The nexus object may refer to any one or all of the following relationships:~~

~~a) One SCSI initiator port to one SCSI target port (an I_T nexus);~~
~~b) One SCSI initiator port to one SCSI target port to one logical unit (an I_T_L nexus);~~
~~c) one SCSI initiator port to one SCSI target port to one logical unit to one task (an I_T_L_Q nexus); or~~
~~d) Either an I_T_L nexus or an I_T_L_Q nexus (denoted as an I_T_L_x nexus).~~

Editor's Note 25: Reason: That is just restating the first sentence

Table 23 ~~maps the nexus object to other identifier objects~~defines the types of nexuses and the identifiers that may be used for each of them.

~~Table 23 — Mapping nexus to SAM-2 identifiers~~Nexus

Nexus [a]	Identifiers ~~contained in nexus~~that may be used	Reference
I_T nexus	Initiator port identifier Target port identifer	~~4.5.9~~4.5.7.2 ~~4.5.14~~4.5.6.2
I_T_L nexus	Initiator port identifier Target port identifer or relative target port identifier Logical unit number	~~4.5.9~~4.5.7.2 ~~4.5.14~~4.5.6.2 and 4.5.5.2 4.5.19.2 and 4.6
I_T_L_Q nexus	Initiator port identifier Target port identifer or relative target port identifier Logical unit number Task tag	~~4.5.9~~4.5.7.2 ~~4.5.14~~4.5.6.2 and 4.5.5.2 4.5.19.2 and 4.6 ~~4.5.23.2~~4.8

[a] I_T_L_x nexus denotes either an I_T_L nexus or an I_T_L_Q nexus.
[b] A nexus may not contain all the identifiers depending on where it is used (e.g., a Nexus attribute in a SCSI Initiator Port object does not contain the initiator port identifier of that object, and the Nexus Attribute in a SCSI Target Port object does not contain the target port identifier or relative target port identifier).

Editor's Note 26: Reason: On the target side, relative target port identifiers are more likely used than the full target port identifier

4.8 Task tag [moved from an attribute in the target Task class]

~~A~~The ~~Task Tag~~task tag identifies a command. The I_T_L_Q nexus representing a task includes a task tag, allowing ~~many~~multiple uniquely identified tagged tasks to be present in a single task set. ~~A task tag is composed of up to 64 bits.~~Each SCSI transport protocol defines the size of the task tag used by SCSI ports using that SCSI transport protocol.

A SCSI initiator device assigns task tag values for each I_T_L_Q nexus in a way that ensures that the nexus uniqueness requirements stated in this subclause are met. ~~Transport~~SCSI transport protocols may define additional restrictions on task tag assignment (e.g., ~~restricting task tag length,~~ requiring task tags to be unique per I_T nexus or per I_T_L nexus, or sharing task tag values with other uses such as task management functions).

An I_T_L_Q nexus that is in use (i.e., during the interval bounded by the events specified in 5.5) shall be unique as seen by the SCSI initiator port originating the command and the logical unit to which the command was addressed, otherwise an overlapped command condition exists (see 5.8.3). An I_T_L_Q nexus is unique if one or more of its components is unique within the specified time interval.

A SCSI initiator device shall not create more than one task from a specific SCSI initiator port having identical values for the target port identifier, logical unit number, and task tag.

Editor's Note 27: Nexus and task tag are not defined as UML classes/objects (yet). Nexus is an attribute in the Task class (based on changes from 07-263). Since UML encourages attributes to be promoted into classes if they are used by multiple classes, and since nexus is used by both the initiator and the target, it is a good candidate for such an upgrade. Task tag is just a part of the nexus, so might be considered an attribute of the Nexus class class.

5 SCSI command model

5.1 The Execute Command procedure call

An application client requests the processing of a command by invoking the SCSI transport protocol services described in 5.4, the collective operation of which is modeled in the following procedure call:

Service Response = Execute Command (IN (I_T_L_Q Nexus, CDB, Task Attribute, [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [Command Reference Number], [Task Priority]),
OUT ([Data-In Buffer], [Sense Data], [Sense Data Length], Status, [Retry Delay Timer]))

Editor's Note 28: We could wipe out this section and the entire concept of "procedure calls" and move the contents into the UML operation descriptions.

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the task (see 4.12).
CDB: Command descriptor block (see 5.2).
Task Attribute: A value specifying one of the task attributes defined in 8.6. SCSI transport protocols may or may not provide the ability to specify a different task attribute for each task (see 8.6.1). For a task that processes linked commands, the Task Attribute shall be the value specified for the first command in a series of linked commands. The Task Attribute specified for the second and subsequent commands shall be ignored.
Data-In Buffer Size: The number of bytes available for data transfers to the Data-In Buffer (see 5.4.3). SCSI transport protocols may interpret this argument to include both the size and the location of the Data-In Buffer.
Data-Out Buffer: A buffer containing command specific information to be sent to the logical unit (e.g., data or parameter lists needed to process the command). The buffer size is indicated by the Data-Out Buffer Size argument. The content of the Data-Out Buffer shall not change during the lifetime of the command (see 5.5) as viewed by the application client.
Data-Out Buffer Size: The number of bytes available for data transfers from the Data-Out Buffer (see 5.4.3).
Command Reference Number (CRN): When this argument is used, all sequential commands of an I_T_L nexus shall include a CRN argument that is incremented by one. The CRN shall be set to one for each I_T_L nexus involving the SCSI port after the SCSI port receives a hard reset or detects I_T nexus loss. The CRN shall be set to one after it reaches the maximum CRN value supported by the protocol. The CRN value zero shall be reserved for use as defined by the SCSI transport protocol. It is not an error for the application client to provide this argument when CRN is not supported by the SCSI transport protocol or logical unit.
Task Priority: The priority assigned to the task (see 8.7).

Output arguments:

Data-In Buffer: A buffer to contain command specific information returned by the logical unit by the time of command completion. The Execute Command procedure call shall not return a status of GOOD, CONDITION MET, INTERMEDIATE, or INTERMEDIATE-CONDITION MET unless the buffer contents are valid. The application client shall treat the buffer contents as invalid unless the command

completes with a status of GOOD, CONDITION MET, INTERMEDIATE, or INTERMEDIATE-CONDITION MET. While some valid data may be present for other values of status, the application client should rely on additional information from the logical unit (e.g., sense data) to determine the state of the buffer contents. If the command ends with a service response of SERVICE DELIVERY OR TARGET FAILURE, the application client shall consider this argument to be undefined.

Sense Data: A buffer containing sense data returned in the same I_T_L_Q nexus transaction (see 3.1.47) as a CHECK CONDITION status (see 5.8.6). The buffer length is indicated by the Sense Data Length argument. If the command ends with a service response of SERVICE DELIVERY OR TARGET FAILURE, the application client shall consider this argument to be undefined.

Sense Data Length: The length in bytes of the Sense Data.

Status: A one-byte field containing command completion status (see 5.3). If the command ends with a service response of SERVICE DELIVERY OR TARGET FAILURE, the application client shall consider this argument to be undefined.

Retry Delay Timer: Additional information about the indicated status code (see 5.3.2).

Service Response assumes one of the following values:

TASK COMPLETE: A logical unit response indicating that the task has ended. The Status argument shall have one of the values specified in 5.3.

ILLEGAL ARGUMENT: The command has been ended due to an unsupported input argument (e.g., CDB too long). All output arguments are invalid.

SERVICE DELIVERY OR TARGET FAILURE: The command has been ended due to a service delivery failure (see 3.1.120) or SCSI target device malfunction. All output ~~parameters~~arguments are invalid.

Editor's Note 29: Reason: If an application client provides a task priority but the transport protocol does not support it, the initiator port should be able to reject the request rather than just ignore the problem.

The SCSI transport protocol events corresponding to a response of TASK COMPLETE or SERVICE DELIVERY OR TARGET FAILURE shall be specified in each SCSI transport protocol standard.

...

5.2 Command descriptor block

The CDB defines the operation to be performed by the device server. CDB formats are defined in SPC-4.

For all commands, if the logical unit detects an invalid parameter in the CDB, then the logical unit shall not process the command.

All CDBs shall have an OPERATION CODE field as the first byte.

Some operation codes provide for modification of their operation based on a service action. In such cases, the combination of operation code value and service action code value may be modeled as a single, unique command determinate. The location of the SERVICE ACTION field in the CDB varies depending on the operation code value.

All CDBs shall contain a CONTROL byte (see table 24). The location of the CONTROL byte within a CDB depends on the CDB format (see SPC-3).

5.3.2 Retry delay timer codes

The retry delay timer codes are specified in table 26 and provide additional information about the reason for the status code.

[table 26]

5.4 SCSI transport protocol services in support of Execute Command

Editor's Note 30: We could wipe out this section and the entire concept of "transport protocol services" and move the contents into the UML operation descriptions.

5.4.1 Overview

The SCSI transport protocol services that support the Execute Command procedure call are described in 5.4. Two groups of SCSI transport protocol services are described. The SCSI transport protocol services that support the delivery of the command and status are described in 5.4.2. The SCSI transport protocol services that support the data transfers associated with processing a command are described in 5.4.3.

5.4.2 Command and Status SCSI transport protocol services

5.4.2.1 Command and Status SCSI transport protocol services overview

All SCSI transport protocol standards shall define the SCSI transport protocol specific requirements for implementing the **Send SCSI Command** request (see 5.4.2.2), the **SCSI Command Received** indication (see 5.4.2.3), the **Send Command Complete** response (see 5.4.2.4), and the **Command Complete Received** confirmation (see 5.4.2.5) SCSI transport protocol services.

All SCSI initiator devices shall implement the **Send SCSI Command** request and the **Command Complete Received** confirmation SCSI transport protocol services as defined in the applicable SCSI transport protocol standards.

All SCSI target devices shall implement the **SCSI Command Received** indication and the **Send Command Complete** response SCSI transport protocol services as defined in the applicable SCSI transport protocol standards.

5.4.2.2 Send SCSI Command transport protocol service request

An application client uses the Send SCSI Command transport protocol service request to request that a SCSI initiator port send a SCSI command.

Send SCSI Command transport protocol service request:

Send SCSI Command (IN (I_T_L_Q Nexus, CDB, Task Attribute, [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [CRN], [Task Priority], [First Burst Enabled]))

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the task (see 4.12).
CDB: Command descriptor block (see 5.2).
Task Attribute: A value specifying one of the task attributes defined in 8.6. For specific requirements on the Task Attribute argument see 5.1.
Data-In Buffer Size: The number of bytes available for data transfers to the Data-In Buffer (see 5.4.3). SCSI transport protocols may interpret the Data-In Buffer Size to include both the size and the location of the Data-In Buffer.
Data-Out Buffer: A buffer containing command specific information to be sent to the logical unit (e.g., data or parameter lists needed to process the command (see 5.1)). The content of the Data-Out Buffer shall not change during the lifetime of the command (see 5.5) as viewed by the application client.
Data-Out Buffer Size: The number of bytes available for data transfers from the Data-Out Buffer (see 5.4.3).
CRN: When CRN is used, all sequential commands of an I_T_L nexus shall include a CRN argument that is incremented by one (see 5.1).
Task Priority: The priority assigned to the task (see 8.7).
First Burst Enabled: An argument specifying that a SCSI transport protocol specific number of bytes from the Data-Out Buffer shall be delivered to the logical unit without waiting for the device server to

invoke the Receive Data-Out SCSI transport protocol service.

5.4.2.3 SCSI Command Received transport protocol service indication

A SCSI target port uses the SCSI Command Received transport protocol service indication to notify a device server that it has received a SCSI command.

SCSI Command Received transport protocol service indication:

SCSI Command Received (IN (I_T_L_Q Nexus, CDB, Task Attribute, [CRN], [Task Priority], [First Burst Enabled]))

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the task (see 4.12).
CDB: Command descriptor block (see 5.2).
Task Attribute: A value specifying one of the task attributes defined in 8.6. For specific requirements on the Task Attribute argument see 5.1.
CRN: When a CRN argument is used, all sequential commands of an I_T_L nexus shall include a CRN argument that is incremented by one (see 5.1).
Task Priority: The priority assigned to the task (see 8.7).
First Burst Enabled: An argument specifying that a SCSI transport protocol specific number of bytes from the Data-Out Buffer are being delivered to the logical unit without waiting for the device server to invoke the Receive Data-Out SCSI transport protocol service.

5.4.2.4 Send Command Complete transport protocol service response

A device server uses the Send Command Complete transport protocol service response to request that a SCSI target port transmit command complete information.

Send Command Complete transport protocol service response:

Send Command Complete (IN (I_T_L_Q Nexus, [Sense Data], [Sense Data Length], Status, Service Response, [Retry Delay Timer]))

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the task (see 4.12).
Sense Data: If present, a Sense Data argument instructs the SCSI target port to return sense data to the SCSI initiator port (see 5.8.6).
Sense Data Length: The length in bytes of the sense data to be returned to the SCSI initiator port.
Status: Command completion status (see 5.1).
Service Response: Possible service response information for the command (see 5.1).
Retry Delay Timer: The Retry Delay Timer code for the command (see 5.3.2).

5.4.2.5 Command Complete Received transport protocol service confirmation

A SCSI initiator port uses the Command Complete Received transport protocol service confirmation to notify an application client that it has received command complete information.

Command Complete Received transport protocol service confirmation:

Command Complete Received (IN (I_T_L_Q Nexus, [Data-In Buffer], [Sense Data], [Sense Data Length], Status, Service Response, [Retry Delay Timer]))

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the task (see 4.12).
Data-In Buffer: A buffer containing command specific information returned by the logical unit on command completion (see 5.1).
Sense Data: Sense data returned in the same I_T_L_Q nexus transaction (see 3.1.47) as a CHECK CONDITION status (see 5.8.6).
Sense Data Length: The length in bytes of the received sense data.
Status: Command completion status (see 5.1).

Service Response: Service response for the command (see 5.1).
Retry Delay Timer: The Retry Delay Timer code for the command (see 5.3.2).

5.4.3 Data transfer SCSI transport protocol services

5.4.3.1 Introduction

The data transfer services described in 5.4.3 provide mechanisms for moving data to and from the SCSI initiator port in response to commands transmitted using the Execute Command procedure call. All SCSI transport protocol standards shall define the protocols required to implement these services.

The application client's Data-In Buffer and/or Data-Out Buffer each appears to the device server as a single, logically contiguous block of memory large enough to hold all the data required by the command (see figure 38). This standard allows either unidirectional or bidirectional data transfer. The processing of a command may require the transfer of data from the application client using the Data-Out Buffer, or to the application client using the Data-In Buffer, or both to and from the application client using both the Data-In Buffer and the Data-Out Buffer.

...

5.4.3.4 Terminate Data Transfer service

5.4.3.4.1 Terminate Data Transfer service overview

~~The terminate data transfer request and confirmation may be used by a task manager to terminate partially completed transfers to the Data-In Buffer or from the Data-Out Buffer.~~

The Terminate Data Transfer SCSI transport protocol service allows a device server or task manager to specify that one or more Send Data-In or Receive Data-Out SCSI transport protocol service requests be terminated by a SCSI target port.

Editor's Note 31: Reason: The second paragraph seems complete as long as task manager is included.

5.4.3.4.2 Terminate Data Transfer transport protocol service request

A device server or task manager uses the Terminate Data Transfer transport protocol service request to request that a SCSI target port terminate data transfers.

Editor's Note 32: Reason: The task manager calls this when aborting tasks

---

Terminate Data Transfer transport protocol service request:

Terminate Data Transfer (IN (Nexus))

Input argument:

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.7).

The SCSI target port terminates all transfer service requests for the specified nexus (e.g., if an I_T_L nexus is specified, then the SCSI target port terminates all transfer service requests from the logical unit for the specified SCSI initiator port).

5.4.3.4.3 Data Transfer Terminated transport protocol service confirmation

A SCSI target port uses the Data Transfer Terminated transport protocol service confirmation to notify a device server or task manager that it has terminated all outstanding data transfers for a specified nexus.

Data Transfer Terminated transport protocol service confirmation:

Data Transfer Terminated (IN (Nexus))

Input argument:

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.7).

This confirmation is returned in response to a Terminate Data Transfer request whether or not the specified nexus existed in the SCSI target port when the request was received. After a Data Transfer Terminated SCSI transport protocol service confirmation has been sent in response to a Terminate Data Transfer SCSI transport protocol service request, the SCSI target port shall not invoke Data-In Delivered or Data-Out Received SCSI transport protocol service confirmations ~~shall not be sent~~ for the tasks specified by the nexus.

7 Task management functions

7.1 ~~Introduction~~Task management function procedure calls

An application client requests the processing of a task management function by invoking the SCSI transport protocol services described in 7.12, the collective operation of which is modeled in the following procedure call format:

~~**Service Response = Function name (IN (nexus), OUT ([additional response information])**~~
Service Response = Function Name (IN (Nexus), OUT ([Additional Response Information])

where:

Function Name	is one of the task management function names listed in table 34
Nexus	is either: a) an I_T Nexus argument; b) an I_T_L Nexus Argument; or c) an I_T_L_Q Nexus argument
Fence	is the Fence input argument described below
Additional Response Information	is the Additional Response Information output argument described below

The task management function names are summarized in table 34.

[table 34]

Input arguments:

~~**Nexus**: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.7) identifying the task or tasks affected by the task management function.~~
~~**I_T Nexus**: A SCSI initiator port and SCSI target port nexus (see 4.7).~~
~~**I_T_L Nexus**: A SCSI initiator port, SCSI target port, and logical unit nexus (see 4.7).~~
~~**I_T_L_Q Nexus**: A SCSI initiator port, SCSI target port, logical unit, and task tag nexus (see 4.7).~~
I_T Nexus: The I_T nexus (see 4.7) affected by the task management function.
I_T_L Nexus: The I_T_L nexus (see 4.7) affected by the task management function.
I_T_L_Q Nexus: The I_T_L_Q nexus (see 4.7) affected by the task management function.

Output arguments:

Additional Response Information: If supported by the transport protocol and the logical unit, then three bytes that are returned along with the service response for certain task management functions (e.g., QUERY UNIT ATTENTION). Transport protocols may or may not support the Additional Response Information argument. A transport protocol supporting the Additional Response Information argument may or may not require that logical units accessible through a target port using that transport protocol support the Additional Response Information argument..

~~One of the following SCSI transport protocol specific responses shall be returned:~~

Service Response assumes one of the following values:

FUNCTION COMPLETE: A task manager response indicating that the requested function is complete. Unless another response is required, the task manager shall return this response upon completion of a task management request supported by the logical unit or SCSI target device to which the request was directed.

FUNCTION SUCCEEDED: An optional task manager response indicating that the requested function is supported and completed successfully. This task manager response shall only be used by functions that require notification of success (e.g., QUERY TASK).

FUNCTION REJECTED: An task manager response indicating that the requested function is not supported by the logical unit or SCSI target device to which the function was directed.

INCORRECT LOGICAL UNIT NUMBER: An optional task router response indicating that the function requested processing for an incorrect logical unit number.

ILLEGAL ARGUMENT: The command has been ended due to an unsupported input argument. All output arguments are invalid.

SERVICE DELIVERY OR TARGET FAILURE: The request was terminated due to a service delivery failure (see 3.1.120) or SCSI target device malfunction. The task manager may or may not have successfully performed the specified function. All output arguments are invalid.

Each SCSI transport protocol standard shall define the events comprising each of these service responses.

The task manager response to task management requests is subject to the presence of access restrictions, as managed by ACCESS CONTROL OUT and ACCESS CONTROL IN commands (see SPC-3), as follows:

a) A task management request of ABORT TASK, ABORT TASK SET, CLEAR ACA, I_T NEXUS RESET, or QUERY TASK shall not be affected by the presence of access restrictions;

b) A task management request of CLEAR TASK SET or LOGICAL UNIT RESET received from a SCSI initiator port that is denied access to the logical unit, either because it has no access rights or because it is in the pending-enrolled state, shall not cause any changes to the logical unit; and c) The task management function service response shall not be affected by the presence of access restrictions.

7.2 ABORT TASK

~~Request~~Procedure call:

Service Response = ABORT TASK (IN (I_T_L_Q Nexus))

Description:

This function shall be supported by all logical units.

The task manager shall abort the specified task, if any, as described in 5.6.2. Previously established conditions, including MODE SELECT parameters, reservations, and ACA shall not be changed by the ABORT TASK function.

A response of FUNCTION COMPLETE shall indicate that the task was aborted or was not in the task set. In either case, the SCSI target device shall guarantee that no further requests or responses are sent from the task. All SCSI transport protocol standards shall support the ABORT TASK task management function.

7.3 ABORT TASK SET

~~Request~~Procedure call:

Service Response = ABORT TASK SET (IN (I_T_L Nexus))

Description:

This function shall be supported by all logical units.

The task manager shall abort all tasks in the task set that were received on the specified I_T nexus as described in 5.6. Tasks received on other I_T nexuses or in other task sets shall not be aborted. This task management function performed is equivalent to a series of ABORT TASK requests.

Other previously established conditions, including MODE SELECT parameters, reservations, and ACA shall not be changed by the ABORT TASK SET function.

All SCSI transport protocol standards shall support the ABORT TASK SET task management function.

7.4 CLEAR ACA

~~Request~~Procedure call:

Service Response = CLEAR ACA (IN (I_T_L Nexus))

Description:

This function shall be supported by a logical unit if it supports ACA (see 5.2).

For the CLEAR ACA task management function, the task set shall be the one defined by the TST field in the Control mode page (see SPC-3).

An application client requests a CLEAR ACA using the faulted I_T nexus (see 3.1.38) to clear an ACA condition from the task set serviced by the logical unit. The state of all tasks in the task set shall be modified as described in 8.8. For a task with the ACA task attribute (see 8.6.5) receipt of a CLEAR ACA function shall have the same effect as receipt of an ABORT TASK function (see 7.2) specifying that task. If successful, this function shall be terminated with a service response of FUNCTION COMPLETE.

If the task manager clears the ACA condition, any task within that task set may be completed subject to the requirements for task set management specified in clause 8.

The service response for a CLEAR ACA request received from an I_T nexus other than the faulted I_T nexus shall be FUNCTION REJECTED.

All SCSI transport protocol standards shall support the CLEAR ACA task management function.

7.5 CLEAR TASK SET

~~Request~~Procedure call:

Service Response = CLEAR TASK SET (IN (I_T_L Nexus))

Description:

This function shall be supported by logical units.

For the CLEAR TASK SET task management function, the task set shall be the one defined by the TST field in the Control mode page (see SPC-3).

All tasks in the task set shall be aborted as described in 5.6.

All pending status and sense data for the task set shall be cleared. Other previously established conditions,including MODE SELECT parameters, reservations, and ACA shall not be changed by the CLEAR TASK SET function.

All SCSI transport protocol standards shall support the CLEAR TASK SET task management function.

7.6 I_T NEXUS RESET

~~Request~~Procedure call:

Service Response = I_T NEXUS RESET (IN (I_T Nexus))

Description:

SCSI transport protocols may or may not support I_T NEXUS RESET and may or may not require logical units accessible through SCSI target ports using such transport protocols to support I_T NEXUS RESET.

Each logical unit accessible through the SCSI target port shall perform the I_T nexus loss functions specified in 6.3.4 for the I_T nexus on which the function request was received, then the SCSI target device shall return a FUNCTION COMPLETE response. After returning a FUNCTION COMPLETE response, the logical unit(s) and the SCSI target port shall perform any additional functions specified by the SCSI transport protocol.

7.7 LOGICAL UNIT RESET

~~Request~~Procedure call:

Service Response = LOGICAL UNIT RESET (IN (I_T_L Nexus))

Description:

This function shall be supported by all logical units.

Before returning a FUNCTION COMPLETE response, the logical unit shall perform the logical unit reset functions specified in 6.3.3.

NOTE 11 - Previous versions of this standard only required LOGICAL UNIT RESET support in logical units that supported hierarchical logical units.

All SCSI transport protocol standards shall support the LOGICAL UNIT RESET task management function.

7.8 QUERY TASK

~~Request~~Procedure call:

Service Response = QUERY TASK (IN (I_T_L_Q Nexus))

Description:

SCSI transport protocols may or may not support QUERY TASK and may or may not require logical units accessible through SCSI target ports using such transport protocols to support QUERY TASK.

The task manager in the specified logical unit shall:

a) if the specified task is present in the task set, then return a service response set to FUNCTION SUCCEEDED; and
b) if the specified task is not present in the task set, then return a service response set to FUNCTION COMPLETE.

7.9 QUERY TASK SET

~~Request~~Procedure call:

Service Response = QUERY TASK SET (IN (I_T_L Nexus))

Description:

SCSI transport protocols may or may not support QUERY TASK SET and may or may not require logical units accessible through SCSI target ports using such transport protocols to support QUERY TASK SET.

The task manager in the specified logical unit shall:

a) if there is any task present in the task set from the specified I_T nexus, then return a service response set to FUNCTION SUCCEEDED; and
b) if there is no task present in the task set from the specified I_T nexus, then return a service response set to FUNCTION COMPLETE.

7.10 QUERY UNIT ATTENTION

~~Request~~Procedure call:

Service Response = QUERY UNIT ATTENTION (IN (I_T_L Nexus), OUT ([Additional Response Information]))

Description:

A SCSI transport protocol may or may not support QUERY UNIT ATTENTION. A SCSI transport protocol supporting QUERY UNIT ATTENTION may or may not require logical units accessible through SCSI target ports using that transport protocol to support QUERY UNIT ATTENTION.

The task manager in the specified logical unit shall:

a) if there is a unit attention condition (see 5.8.7) or a deferred error (see SPC-3) pending for the specified I_T nexus, then return a service response set to FUNCTION SUCCEEDED; and
b) if there is no unit attention condition or deferred error pending for the specified I_T nexus, then return a service response set to FUNCTION COMPLETE.

If the service response is not FUNCTION SUCCEEDED, then the task manager shall set the Additional Response Information argument to 000000h.

If the service response is FUNCTION SUCCEEDED, the task manager shall set the Additional Response Information argument as defined in table 35.

7.11 Task management function lifetime

The task manager shall create a task management function upon receiving a Task Management Request Received indication (see 7.12). The task management function shall exist until:

...

7.12 Task management SCSI transport protocol services

Editor's Note 33: We could wipe out this section and the entire concept of transport protocol services and move the contents into the UML operation descriptions.

7.12.1 Task management SCSI transport protocol services overview

The SCSI transport protocol services described in this subclause are used by a SCSI initiator device and SCSI target device to process a task management procedure call. The following arguments are passed:

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.12).
Function Identifier: Argument encoding the task management function to be performed.

All SCSI transport protocol standards shall define the SCSI transport protocol specific requirements for implementing the **Send Task Management Request** request (see 7.10.2), the **Task Management Request Received** indication (see 7.10.3), the **Task Management Function Executed** response (see 7.10.4), and the **Received Task Management Function Executed** ~~(see 7.10.5)~~ confirmation (see 7.10.5) SCSI transport protocol services.

A SCSI transport protocol standard may specify different implementation requirements for the **Send Task Management Request** request SCSI transport protocol service for different values of the Function Identifier argument.

All SCSI initiator devices shall implement the **Send Task Management Request** request and the **Received Task Management Function Executed** confirmation SCSI transport protocol services as defined in the applicable SCSI transport protocol standards.

All SCSI target devices shall implement the **Task Management Request Received** indication and the **Task Management Function Executed** response SCSI transport protocol services as defined in the applicable SCSI transport protocol standards.

7.12.2 Send Task Management Request transport protocol service request

An application client uses the Send Task Management Request transport protocol service request to request that a SCSI initiator port send a task mangement request.

Send Task Management Request transport protocol service request:

Send Task Management Request (IN (Nexus, Function Identifier))

Input arguments:

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.12).
Function Identifier: Argument encoding the task management function to be performed.

7.12.3 Task Management Request Received transport protocol service indication

A SCSI target port uses the Task Management Request Received transport protocol service indication to notify a task manager that it has received a task management request.

Task Management Request Received transport protocol service indication:

Task Management Request Received (IN (Nexus, Function Identifier))

Input arguments:

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.12).
Function Identifier: Argument encoding the task management function to be performed.

7.12.4 Task Management Function Executed transport protocol service response

A task manager uses the ~~Send Task Management Request~~Task Management Function Executed transport protocol service response to request that a SCSI target port transmit task management function executed information.

Task Management Function Executed transport protocol service response:

Task Management Function Executed (IN (Nexus, Service Response, [Additional Response Information]))

Input arguments:

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.12).
Service Response: An encoded value representing one of the following:
FUNCTION COMPLETE: The requested function has been completed.
FUNCTION SUCCEEDED: The requested function is supported and completed successfully.
FUNCTION REJECTED: The task manager does not implement the requested function.
INCORRECT LOGICAL UNIT NUMBER: An optional task router response indicating that the function requested processing for an incorrect logical unit number.
SERVICE DELIVERY OR TARGET FAILURE: The request was terminated due to a service delivery failure (see 3.1.112) or SCSI target device malfunction. The task manager may or may not have successfully performed the specified function.
Additional Response Information: The Additional Response Information output argument for the task management procedure call (see 7.1):

7.12.5 Received Task Management Function Executed transport protocol service confirmation

A SCSI initiator port uses the Received Task Management Function Executed transport protocol service confirmation to notify an application client that it has received task management function executed information.

Received Task Management Function Executed transport protocol service confirmation:

Received Task Management Function Executed (IN (Nexus, Service Response, [Additional Response Information]))

Input arguments:

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.12).
Service Response: An encoded value representing one of the following:
FUNCTION COMPLETE: The requested function has been completed.
FUNCTION SUCCEEDED: The requested function is supported and completed successfully.
FUNCTION REJECTED: The task manager does not implement the requested function.
INCORRECT LOGICAL UNIT NUMBER: An optional task router response indicating that the function requested processing for an incorrect logical unit number.
SERVICE DELIVERY OR TARGET FAILURE: The request was terminated due to a service delivery failure

(see 3.1.112) or SCSI target device malfunction. The task manager may or may not have successfully performed the specified function.

Additional Response Information: The Additional Response Information output argument for the task management procedure call (see 7.1):

Each SCSI transport protocol shall allow a **Received Task Management Function Executed** confirming completion of the requested task to be associated with the corresponding **Send Task Management Request**.

8.6 Task attributes

The application client shall assign a task attribute (see table 38) to each task.

SCSI transport protocols shall provide the capability to specify a unique task attribute for each task.