Date: September 14, 2007

To: T10 Committee (SCSI)

From: George Penokie (IBM)

Subject: SAM-4: Changes requested from 03/2007 editing session

1 Overview

During the SAM-4 editing session in March it was requested that the UML diagrams be enhanced to include all the parameters that are listed in the Execute Command procedure call and to add in associations between some of the classes in the SCSI initiator device and the logical unit class diagrams. This proposal makes those changes.

A change was made to rename task tag to task identifier. If accepted that requires 2 changes in SPC-4 and 2 changes in SAS-2.

Revision 2 is a major rewrite of what was in revsion 1.

3.1 Definitions

3.1.1 ACA command: A command performed by a task with the ACA task attribute (see 3.1.8, and 8.6.5).

3.1.2 additional sense code: A combination of the ADDITIONAL SENSE CODE field and the ADDITIONAL SENSE CODE QUALIFIER field in the sense data (see 3.1.108 and SPC-3).

3.1.3 aggregation: When referring to classes (see 3.1.13), a form of association that defines a whole-part relationship between the whole (i.e., aggregate) and its parts.

3.1.4 application client: A~~n~~ ~~object~~ class whose objects are, or an object that is, the source of commands and task management function requests. See 4.5.10x.

3.1.5 argument: A datum provided as input to or output from a procedure call (see 3.1.80).

3.1.6 association: When referring to classes (see 3.1.13), a relationship between two or more classes that specifies connections among their objects (i.e., a relationship that specifies that objects of one class are connected to objects of another class).

3.1.7 attribute: When referring to classes (see 3.1.13), a named property of a class that describes the range of values that the class or its objects may hold. When referring to objects (see 3.1.71), a named property of the object.

3.1.8 auto contingent allegiance (ACA): The task set condition established following the return of a CHECK CONDITION status when the NACA bit is set to one in the CONTROL byte. See 5.8.2 .

3.1.9 background operation: An operation started by a command that continues processing after the task containing the command is no longer in the task set. See 5.5 .

3.1.10 blocked task state: When in this state a task is prevented from completing due to an ACA condition.

3.1.11 blocking boundary: A task set boundary denoting a set of conditions that inhibit tasks outside the boundary from entering the enabled task state.

3.1.12 byte: An 8-bit construct.

3.1.13 class: A description of a set of objects that share the same attributes, operations, relationships (e.g., aggregation, association, generalization, and dependency), and semantics. Classes may have attributes and may support operations.

3.1.14 class diagram: Shows a set of classes and their relationships. Class diagrams are used to illustrate the static design view of a system.

3.1.15 client-server: A relationship established between a pair of distributed entities where one (the client) requests the other (the server) to perform some operation or unit of work on the client's behalf.

3.1.16 client: An entity that requests a service from a server. This standard defines one client, the application client.

3.1.17 code value: A defined numeric value, possibly a member of a series of defined numeric values, representing an identified and described instance or condition. Code values are defined to be used in a specific field (see 3.1.42), in a procedure call input argument (see 3.6.4), in a procedure call output argument, or in a procedure call result.

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

3.1.19 command descriptor block (CDB): A structure used to communicate a command from an application client to a device server. A CDB may have a fixed length of 6 bytes, 10 bytes, 12 bytes, or 16 bytes, or a variable length of between 12 and 260 bytes. See 5.2 and SPC-3.

3.1.20 command standard: A SCSI standard that defines the model, commands, and parameter data for a device type (e.g., SPC-3, SBC-2, SSC-2, SMC-2, MMC-4, or SES-2). See clause 1.

3.1.21 completed command: A command that has ended by returning a status and service response of TASK COMPLETE.

3.1.22 completed task: A task that has ended by returning a status and service response of TASK COMPLETE.

3.1.23 confirmation: A response returned to an application client or device server that signals the completion of a service request.

3.1.24 confirmed SCSI transport protocol service: A service available at the SCSI transport protocol service interface that includes a confirmation of completion. See 4.9 .

3.1.25 constraint: When referring to classes (see 3.1.13) and objects (see 3.1.71), a mechanism for specifying semantics or conditions that are maintained as true between entities (e.g., a required condition between associations).

3.1.26 current task: A task that has a data transfer SCSI transport protocol service request in progress (see 5.4.3) or is in the process of sending command status. Each SCSI transport protocol standard may define the SCSI transport protocol specific conditions under which a task is considered a current task.

3.1.27 deferred error: An error generated by a background operation (see SPC-3).

3.1.28 dependency: A relationship between two elements in which a change to one element (e.g., the supplier) may affect or supply information needed by the other element (e.g., the client).

3.1.29 dependent logical unit: A logical unit that is addressed via some other logical unit(s) in a hierarchical logical unit structure (see 3.1.45), also a logical unit that is at a higher numbered level in the hierarchy than the referenced logical unit (see 4.5.19.4).

3.1.30 device model: The description of a type of SCSI target device (e.g., block, stream).

3.1.31 device server: A~~n object~~ class whose objects process, or an object that ~~within a logical unit that~~ processes, SCSI tasks according to the requirements for task management described in clause 8. See x.x.x.

3.1.32 device service request: A request submitted by an application client conveying a command to a device server.

3.1.33 device service response: The response returned to an application client by a device server on completion of a command.

3.1.34 domain: An I/O system consisting of a set of SCSI devices and a service delivery subsystem, where the SCSI devices interact with one another by means of a service delivery subsystem.

3.1.35 dormant task state: When in this state a task is prevented from entering the enabled task state (see 3.1.36) due to the presence of certain other tasks in the task set.

3.1.36 enabled task state: When in this state a task may complete at any time.

3.1.37 extended logical unit addressing: The logical unit addressing method used by special function logical units (e.g., well known logical units). See 4.6.10 .

3.1.38 faulted I_T nexus: The I_T nexus on which a CHECK CONDITION status was returned that resulted in the establishment of an ACA. The faulted I_T nexus condition is cleared when the ACA condition is cleared.

3.1.39 faulted task set: A task set that contains a faulting task. The faulted task set condition is cleared when the ACA condition resulting from the CHECK CONDITION status is cleared.

3.1.40 faulting command: A command that completed with a status of CHECK CONDITION that resulted in the establishment of an ACA.

3.1.41 faulting task: A task that has completed with a status of CHECK CONDITION that resulted in the establishment of an ACA.

3.1.42 field: A group of one or more contiguous bits, part of a larger structure (e.g., a CDB (see 3.1.19) or sense data (see 3.1.108)).

3.1.43 generalization: When referring to classes (see 3.1.13), a relationship among classes where one class (i.e., superclass) shares the attributes and operations on one or more classes (i.e., subclasses).

3.1.44 hard reset: A condition resulting from a power on condition or a reset event in which the SCSI device performs the hard reset operations described in 6.3.2, SPC-3, and the appropriate command standards.

3.1.45 hierarchical logical unit: An inverted tree structure for forming and parsing logical unit numbers (see 3.1.66) containing up to four addressable levels (see 4.5.15).

3.1.46 I_T nexus: A nexus between a SCSI initiator port and a SCSI target port. See 4.7 .

3.1.47 I_T nexus loss: A condition resulting from a hard reset condition or an I_T nexus loss event in which the SCSI device performs the I_T nexus loss operations described in 6.3.4, SPC-3, and the appropriate command standards.

3.1.48 I_T nexus loss event: A SCSI transport protocol specific event that results in an I_T nexus loss condition as described in 6.3.4.

3.1.49 I_T_L nexus: A nexus between a SCSI initiator port, a SCSI target port, and a logical unit (see 4.7).

3.1.50 I_T_L_Q nexus: A nexus between a SCSI initiator port, a SCSI target port, a logical unit, and a task. See 4.7 .

3.1.51 I_T_L_Q nexus transaction: The information transferred between SCSI ports in a single data structure with defined boundaries (e.g., an information unit).

3.1.52 I_T_L_x nexus: Either an I_T_L nexus or an I_T_L_Q nexus. See 4.7 .

3.1.53 I/O operation: An operation defined by a command or a task management function.

3.1.54 implementation specific: A requirement or feature that is defined in a SCSI standard but whose implementation may be specified by the system integrator or vendor.

3.1.55 initiator port identifier: A value by which a SCSI initiator port is referenced within a domain. See 4.5.9 .

3.1.56 initiator port name: A name (see 3.1.68) of a SCSI initiator port that is world wide unique within the SCSI transport protocol of the SCSI domain of that SCSI initiator port (see 4.5.5). The name may be made available to other SCSI devices or SCSI ports in that SCSI domain in SCSI transport protocol specific ways. See 4.5.9

3.1.57 in transit: Information that has been delivered to a service delivery subsystem for transmission, but not yet received.

3.1.58 instance: A concrete manifestation of an abstraction to which a set of operations may be applied and which may have a state that stores the effects of the operation (e.g., an object is an instance of a class).

3.1.59 implicit head of queue: An optional processing model for specified commands wherein a task may be treated as if it had been received with a HEAD OF QUEUE task attribute. See 8.2 .

3.1.60 layer: A subdivision of the architecture constituted by SCSI initiator device and SCSI target device elements at the same level relative to the interconnect.

3.1.61 link: An individual connection between two objects in an object diagram. Represents an instance of an association.

3.1.62 logical unit: A ~~SCSI target device class, containing a device server and task manager,~~ class whose objects implement, or an object that implements, a device model and manages tasks to process commands sent by an application client. See 4.5.19 .

3.1.63 logical unit reset: A condition resulting from a hard reset condition or a logical unit reset event in which the logical unit performs the logical unit reset operations described in 6.3.3, SPC-3, and the appropriate command standards.

3.1.64 logical unit reset event: An event that results in a logical unit reset condition as described in 6.3.3.

3.1.65 logical unit inventory: The list of the logical unit numbers reported by a REPORT LUNS command (see SPC-3).

3.1.66 logical unit number (LUN): A 64-bit or 16-bit identifier for a logical unit. See 4.6.

3.1.67 multiplicity: When referring to classes (see 3.1.13), an indication of the range of allowable instances that a class or an attribute may have.

3.1.68 name: A label of an object that is unique within a specified context and should never change (e.g., the terms name and world wide identifier (WWID) may be interchangeable).

3.1.69 nexus: A relationship between two SCSI devices, and the SCSI initiator port and SCSI target port objects within those SCSI devices. See 4.7 .

3.1.70 non-faulted I_T nexus: An I_T nexus that is not a faulted I_T nexus (see 3.1.38).

3.1.71 object: An entity with a well-defined boundary and identity that encapsulates state and behavior. All objects are instances of classes (see 3.1.58). ~~(i.e., a concrete manifestation of a class is an object)~~.

3.1.72 object diagram: shows a set of objects and their relationships at a point in time. Object diagrams are used to illustrate static shapshots of instances of the things found in class diagrams.

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 query ~~question~~. A request may change the state of the object but a query ~~question~~ should not.

3.1.74 peer entities: Entities within the same layer (see 3.1.60).

3.1.75 pending command: From the point of view of the application client, the description of command between the time that the application client calls the **Send SCSI Command** SCSI transport protocol service and the time one of the SCSI target device responses described in 5.5 is received.

3.1.76 power cycle: Power being removed from and later applied to a SCSI device.

3.1.77 power on: A condition resulting from a power on event in which the SCSI device performs the power on operations described in 6.3.1, SPC-3, and the appropriate command standards.

3.1.78 power on event: Power being applied to a SCSI device, resulting in a power on condition as described in 6.3.1.

3.1.79 procedure: An operation that is invoked through an external calling interface.

3.1.80 procedure call: The model used by this standard for the interfaces involving both the SAL (see 3.1.91) and STPL (see 3.1.102), having the appearance of a programming language function call.

3.1.81 protocol: A specification and/or implementation of the requirements governing the content and exchange of information passed between distributed entities through a service delivery subsystem.

3.1.82 queue: The arrangement of tasks within a task set (see 3.1.127), usually according to the temporal order in which they were created.

3.1.83 receiver: A client or server that is the recipient of a service delivery transaction.

3.1.84 reference model: A standard model used to specify system requirements in an implementation-independent manner.

3.1.85 relative port identifier: An identifier for a SCSI port that is unique within a SCSI device. See 4.5.5.2 .

3.1.86 request: A transaction invoking a service.

3.1.87 request-response transaction: An interaction between a pair of distributed, cooperating entities, consisting of a request for service submitted to an entity followed by a response conveying the result.

3.1.88 reset event: A SCSI transport protocol specific event that results in a hard reset condition as described in 6.3.2.

3.1.89 response: A transaction conveying the result of a request.

3.1.90 role: When referring to classes (see 3.1.13) and objects (see 3.1.71), a label at the end of an association or aggregation that defines a relationship to the class on the other side of the association or aggregation.

3.1.91 SCSI application layer (SAL): The protocols and procedures that implement or issue commands and task management functions by using services provided by a STPL (see 3.1.102).

3.1.92 SCSI device: A class whose objects are, or an object that is, ~~device that contains one or more SCSI ports that are~~ connected to a service delivery subsystem and supports a SCSI application protocol. See x.x.x.

3.1.93 SCSI device name: A name (see 3.1.68) of a SCSI device that is world wide unique within the SCSI transport protocol of a SCSI domain in which the SCSI device has SCSI ports (see 4.5.4.2). The SCSI device name may be made available to other SCSI devices or SCSI ports in SCSI transport protocol specific ways.

3.1.94 SCSI event: A condition defined by this standard (e.g., logical unit reset) that is detected by SCSI device and that requires notification of its occurrence within the SCSI device. See clause 6

3.1.95 SCSI I/O system: An I/O system, consisting of two or more SCSI devices, a SCSI interconnect and a SCSI transport protocol that collectively interact to perform SCSI I/O operations.

3.1.96 SCSI initiator device: A class whose objects originate, or an object that originates, ~~SCSI device containing application clients and SCSI initiator ports that originates~~ device service and task management requests to be processed by a SCSI target device and receives device service and task management responses from SCSI target devices. When used this term refers to SCSI initiator devices. See x.x.x.

3.1.97 SCSI initiator port: A class whose objects act, or an object that acts, ~~SCSI initiator device object that acts~~ as the connection between application clients and a service delivery subsystem through which requests, indications, responses, and confirmations are routed. In all cases when this term is used it refers to an initiator port. See x.x.x.

3.1.98 SCSI port: A class whose objects connect, or an object that connects, ~~SCSI device resident object that connects~~ the application client, device server or task manager to a service delivery subsystem through which requests and responses are routed. SCSI port is synonymous with port. A SCSI port is one of: a SCSI initiator port (see 3.1.97) or a SCSI target port (see 3.1.101). See x.x.x.

3.1.99 SCSI port identifier: A value by which a SCSI port is referenced within a domain. The SCSI port identifier is either an initiator port identifier (see 3.1.55) or a target port identifier (see 3.1.116).

3.1.100 SCSI target device: A class whose objects recieve, or an object that receives, ~~SCSI device containing logical units and SCSI target ports that receives~~ device service and task management requests for processing and sends device service and task management responses to SCSI initiator devices. When used this term refers to SCSI target devices. See x.x.x.

3.1.101 SCSI target port: A class whose objects contian, or an object that contains, ~~SCSI target device object that contains~~ a task router and acts as the connection between device servers and task managers and a service delivery subsystem through which indications and responses are routed. When this term is used it refers to a SCSI target port. See x.x.x.

3.1.102 SCSI transport protocol layer (STPL): The protocol and services used by a SAL (see 3.1.91) to transport data representing a SCSI application protocol transaction.

3.1.103 SCSI transport protocol service confirmation: A procedure call from the STPL notifying the SAL that a SCSI transport protocol service request has completed.

3.1.104 SCSI transport protocol service indication: A procedure call from the STPL notifying the SAL that a SCSI transport protocol transaction has occurred.

3.1.105 SCSI transport protocol service request: A procedure call to the STPL to begin a SCSI transport protocol service transaction.

3.1.106 SCSI transport protocol service response: A procedure call to the STPL containing a reply from the SAL in response to a SCSI transport protocol service indication.

3.1.107 sender: A client or server that originates a service delivery transaction.

3.1.108 sense data: Data returned to an application client in the same I_T_L_Q nexus transaction (see 3.1.51) as a CHECK CONDITION status (see 5.8.6). Fields in the sense data are referenced by name in this standard. See SPC-3 for a complete sense data format definition. Sense data may also be retrieved using the REQUEST SENSE command (see SPC-3).

3.1.109 sense key: The SENSE KEY field in the sense data (see 3.1.108 and SPC-3).

3.1.110 server: An entity that performs a service on behalf of a client.

3.1.111 service: Any operation or function performed by a SCSI object that is invoked by other SCSI objects.

3.1.112 service delivery failure: Any non-recoverable error causing the corruption or loss of one or more service delivery transactions while in transit.

3.1.113 service delivery subsystem: A class whose objects are, or an object that is, ~~That~~ part of a SCSI I/O system that transmits service requests to a logical unit or SCSI target device and returns logical unit or SCSI target device responses to a SCSI initiator device. See x.x.x.

3.1.114 service delivery transaction: A request or response sent through a service delivery subsystem.

3.1.115 standard INQUIRY data: Data returned to an application client as a result of an INQUIRY command with the EVPD bit set to zero. Fields in the standard INQUIRY data are referenced by name in this standard and SPC-3 contains a complete definition of the standard INQUIRY data format.

3.1.116 target port identifier: A value by which a SCSI target port is referenced within a domain. See 4.5.14 .

3.1.117 target port name: A 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 (see 4.5.5). The name may be made available to other SCSI devices or SCSI ports in that SCSI domain in SCSI transport protocol specific ways. See 4.5.14 .

3.1.118 task: A~~n~~ class whose objects are, or an object that is, ~~object~~ within the logical unit representing the work associated with a command. See 4.5.19 .

3.1.119 task attribute: An attribute of a task (see 3.1.118) that specifies the processing relationship of a task with regard to other tasks in the task set (see 3.1.127). See 8.6.

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 identifier ~~tag~~: ~~An attribute of a task class containing up to 64 bits~~ The portion (i.e, Q) of an I_T_L_Q nexus 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 .

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.

3.1.123 task management request: A request submitted by an application client, invoking a task management function to be processed by a task manager.

3.1.124 task management response: The response returned to an application client by a task manager on completion of a task management request.

3.1.125 task manager: A class whose objects are, or an object that is, ~~server~~ 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: A~~n object~~ class whose objects are, or an object that is, within 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 x.x.x.

3.1.127 task set: A class whose objects are, or an object that is, a group of tasks within a logical unit, whose interaction is dependent on the task management (e.g., queuing) and ACA requirements. See x.x.x.

3.1.128 task tag: A term used by previous versions of this standard (see Annex x).

3.1.129 **third-party command:** A command that requires a logical unit within a SCSI target device to assume the SCSI initiator device role and send command(s) to another SCSI target device.

3.1.130 transaction: A cooperative interaction between two entities, involving the exchange of information or the processing of some request by one entity on behalf of the other.

3.1.131 unconfirmed SCSI transport protocol service: A service available at the SCSI transport protocol service interface that does not result in a completion confirmation. See 4.9 .

3.1.132 well known logical unit: A class whose objects are, or an object that is, a logical unit that only performs specific functions. Well known logical units allow an application client to issue requests to receive and manage specific information relating to a SCSI target device. See 4.5.24 .

3.1.133 well known logical unit number (W-LUN): The logical unit number that identifies a well known logical unit. See 4.6.11 .

3.2 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 identifier ~~tag~~) or be a complex entity that may:

a) contain multiple attributes: or
b) perform~~s~~ a set of operations or services on behalf of another class.

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 1, 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 1 — 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.~~

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.

4.4 SCSI classes

4.4.1 SCSI classes overview

Figure 2 shows the main functional classes of the SCSI domain. This standard defines these classes in greater detail.

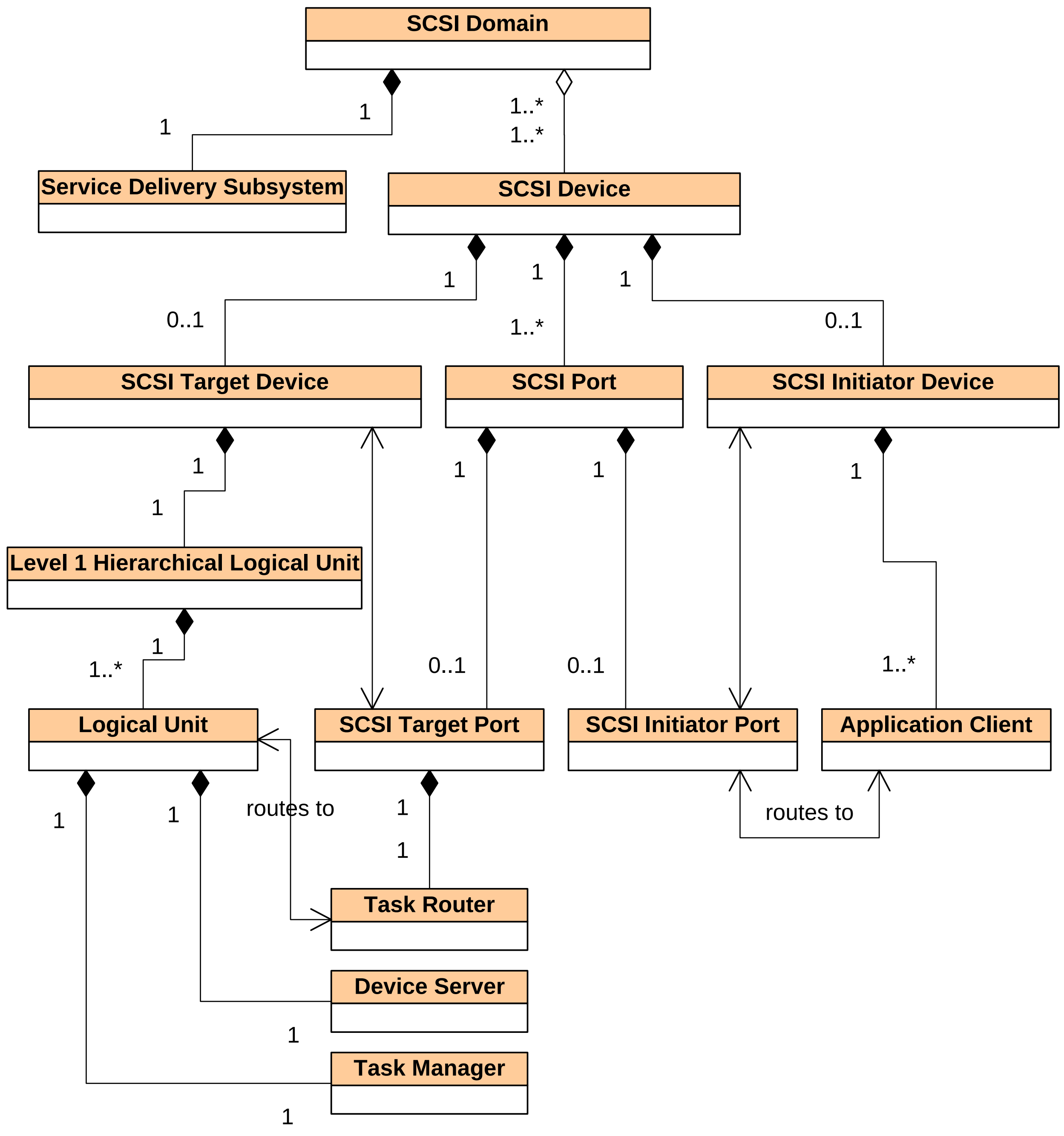

Figure 2 — SCSI Domain class diagram overview

4.4.2 SCSI Port class

4.4.2.1 SCSI Port class overview

~~A~~ The SCSI Port class (see figure 3) contains the:

a) SCSI Target Port class (see 4.4.3) ~~that contains the:~~
   ~~A) Task Router class (see 4.4.5);~~
b) SCSI Initiator Port class (see 4.4.4.1); or
c) both.

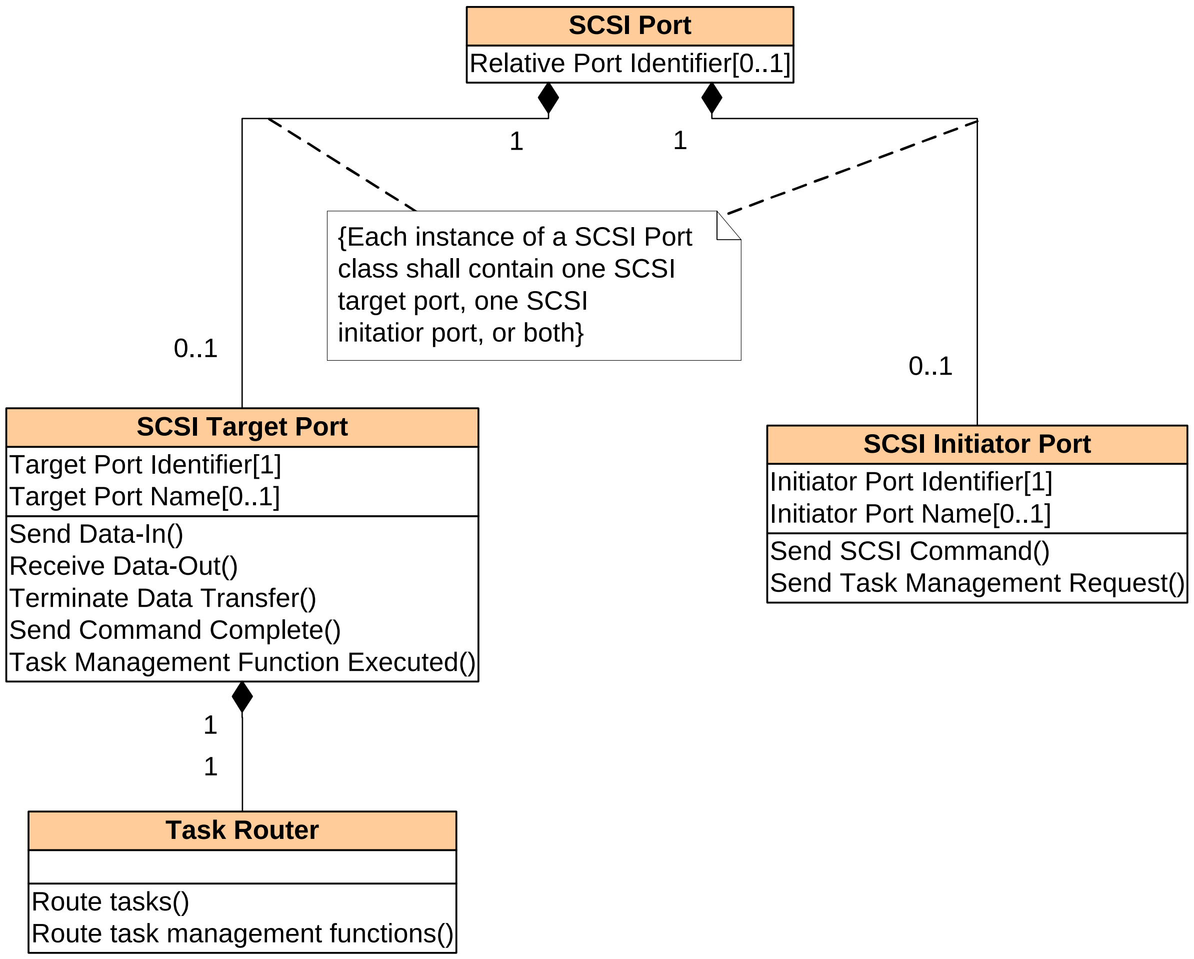
SCSI Port
Relative Port Identifier[0..1]
1
1
{Each instance of a SCSI Port class shall contain one SCSI target port, one SCSI initatior port, or both}
0..1
0..1
SCSI Target Port
Target Port Identifier[1]
Target Port Name[0..1]
Send Data-In()
Receive Data-Out()
Terminate Data Transfer()
Send Command Complete()
Task Management Function Executed()
SCSI Initiator Port
Initiator Port Identifier[1]
Initiator Port Name[0..1]
Send SCSI Command()
Send Task Management Request()
1
1
Task Router
Route tasks()
Route task management functions()

Figure 3 — SCSI Port class diagram

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.

4.4.2.2 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.

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.

4.4.3 SCSI Target Port class

4.4.3.1 SCSI Target Port class overview

~~A~~ The SCSI Target Port class (see figure 3) contains the:

a) Task Router class (see 4.4.5);

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

The SCSI Target Port class processes the:

a) Send Data-in operation (see xx.xx.xx) to send data to the service delivery subsystem;
b) Received Data-out operation (see xx.xx.xx) to receive data from the service delivery subsystem;
c) Terminate Data Transfer operation (see xx.xx.xx) to terminate data transfers;
d) Send Command Complete operation (see xx.xx.xx) to transmit a command complete indication to the service delivery subsystem; and
e) Task Management Function Executed operation (see xx.xx.xx) to transmit a task management function executed indication to the service delivery subsystem.

4.4.3.2 Target Port Identifier attribute

The Target Port Identifier attribute contains a target port identifier (see 3.1.115) for a SCSI target port. The target port identifier is a value by which a SCSI target port is referenced within a domain.

4.4.3.3 Target Port Name attribute

A Target Port Name attribute contains an optional name (see 3.1.67) 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.

4.4.4 SCSI Initiator Port class

4.4.4.1 SCSI Initiator Port class overview

The SCSI Initiator Port class connects SCSI initiator devices to a service delivery subsystem.

The SCSI Initiator Port (see figure 3) class processes the:

a) Send SCSI Command operation (see xx.xx.xx) to send a SCSI command to the service delivery subsystem; and
b) Send Task Management Request operation (see xx.xx.xx) to send a task management request to the service delivery subsystem.

4.4.4.2 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.

4.4.4.3 Initiator Port Name attribute

A Initiator Port Name attribute contains an optional name (see 3.1.67) of a SCSI initiator port that is world wide unique 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.

4.4.5 Task Router class

The Task Router class (see figure 3) routes:

a) ~~information (e.g., commands and~~ task management functions~~)~~ between a ~~logical unit~~ task manager and a service delivery subsystem by processing the Route Task Management Functions operation; and
b) commands between a logical unit task manager and a service delivery subsystem by processing the Route Tasks 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 identifiers ~~tags~~ on commands (see 5.8.3).

4.5.9 SCSI Initiator Device class

The SCSI Initiator Device class (see figure 4) is a SCSI Device class that contains the:

a) Application Client class (see 4.5.10).
a) ~~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: and:~~
   ~~B) application client task class .~~

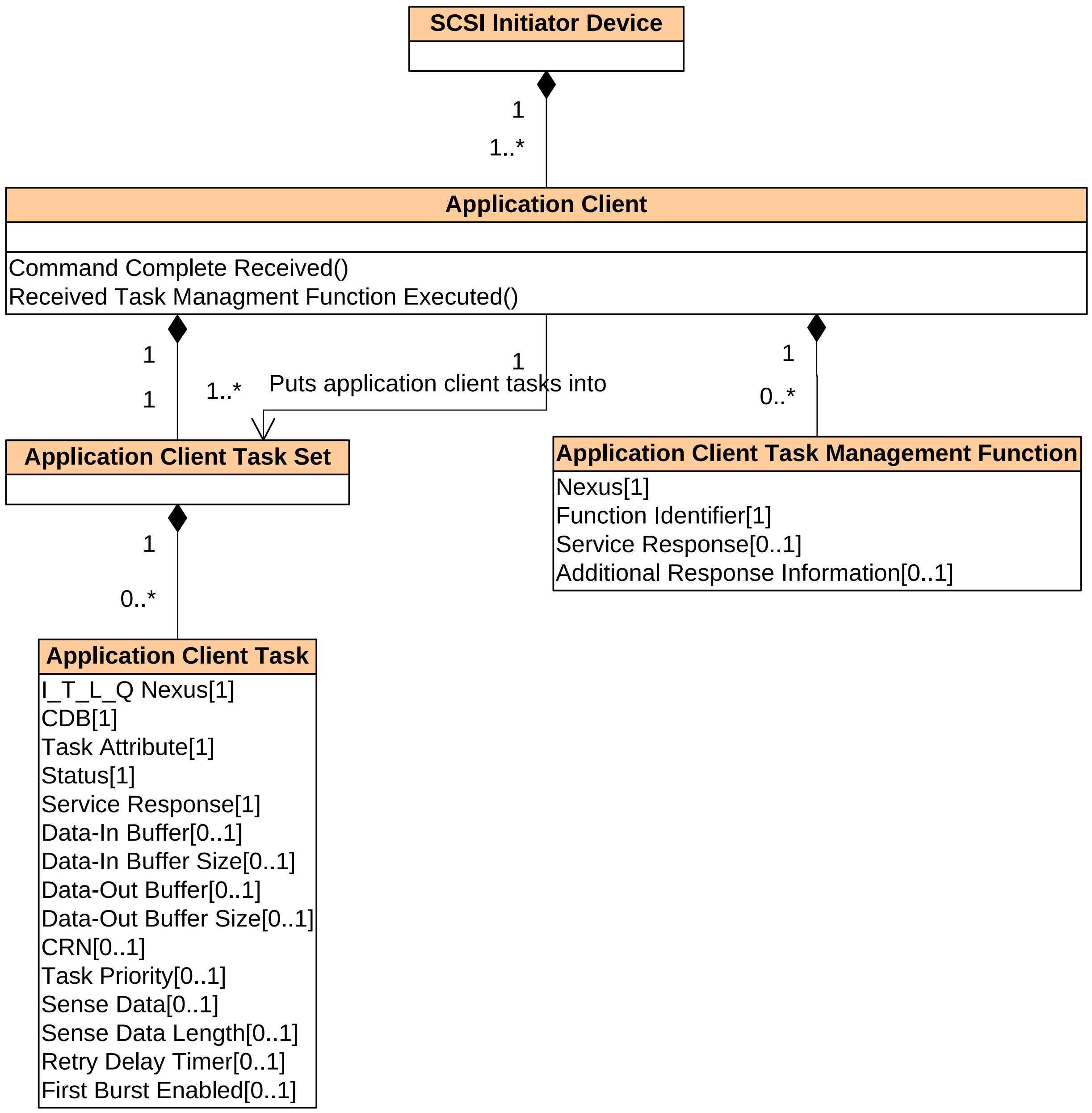

Figure 4 — SCSI initiator device class diagram

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 task management functions; and
  B) one application client task set.

4.5.10 Application Client class

~~An application client class contains zero or more application client tasks.~~

The Application Client class (see figure 4) contains the:

a) Application Client Task Management Function class (see 4.5.11); and
b) Application Client Task Set class (see 4.5.12).

The Application Client class processes the:

a) Command Complete Received operation (see xx.xx.xx) to determine when a requested a command has completed; and
b) Received Task Management Function Executed operation (see xx.xx.xx) to determine when a requested task management function has completed.

An Application Client class originates commands by issuing **Send SCSI Command** requests ~~(see 5.4.2)~~(see 5.4.2). This action creates an application client task object that is placed into the application client task set. The application client task object remains in the application client task set until the appliction client determines when a command that it has originated completes using the command lifetime (see 5.5), including the processing of a Command Complete Received operation.

~~An~~ The Application Client class originates task management requests by issuing Function name service requests ~~(see clause 7)~~(see clause 7). An Application Client class determines when a task management request that it has originated completes using the task management function lifetime information (see 7.11), including the processing of a Received Task Management Function operation.

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

~~An~~ The 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~~7.13.

4.5.11 Application Client Task Management Function class

4.5.11.1 Application Client Task ~~class~~ Management Function class overview

~~An~~ The Application Client Task Management Function class ~~is an application client class that~~ represents a ~~command (see clause 5)~~SCSI task management function (see clause 7).

4.5.11.2 Nexus attribute

The nexus attribute contains the nexus of the task management function (see clause 7).

4.5.11.3 Function Identifier attribute

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

4.5.11.4 Service Response attribute

The Service Response attribute contains the service response (see 5x.1x).

4.5.11.5 Additional response information attribute

The additional response information attribute contains any additional response information for the task management function (see clause 7).

4.5.12 Application Client Task Set class

The Application Client Task Set class (see figure 4) contains the:

a) Application Client Task class (see x.x.x).

Each instance of an Application Client Task Set class shall contain the following objects:

a) zero or more application client tasks.

The interactions among the application client tasks in an application client task set are not specified in this standard.

4.5.13 Application Client Task class

4.5.13.1 Application Client Task class overview

~~An~~ The 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 I_T_L_Q nexus attribute

The I_T_L_Q nexus attribute contains the I_T_L_Q nexus of the task (see 4.7).

The task identifier (i.e., the Q in an I_T_L_Q nexus) represents ~~representing~~ a task ~~includes a task attribute~~, allowing many uniquely identified tagged tasks to be outstanding at once. ~~present in a single task set~~. A task identifier ~~attribute~~ is composed of up to 64 bits.

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

The SCSI initiator device assigns a task ~~attribute~~ identifier value 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 identifier ~~attribute~~ assignments (e.g., restricting task ~~attribute~~ identifier length, requiring task ~~attribute~~ identifier to be unique per I_T nexus or per I_T_L nexus, or sharing task ~~attribute~~ identifier 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.

The 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 identifier ~~attribute~~.

4.5.13.3 CDB attribute

The CDB attribute contains a CDB (see 5.2 and SPC-3) that defines the work to be performed by a logical unit.

4.5.13.4 ~~CDB~~ Task Attribute attribute

~~A~~ The Task Attribute attribute (see 8.6) contains the task attribute (e.g., simple, ordered, head or queue, ACA) of ~~identifies~~ a command.

4.5.13.5 Status attribute

The status attribute contains the status of the completed command (see 5.3).

4.5.13.6 Service response attribute

The service response attribute contains the service response for the completed command (see 5.4.2.5)

4.5.13.7 Data-In buffer attribute

The data-in buffer attribute contains the Data-In Buffer argument from an Execute Command procedure call (see 5.1).

4.5.13.8 Data-In buffer size attribute

The data-in buffer size attribute contains the Data-In Buffer Size argument from an Execute Command procedure call (see 5.1).

4.5.13.9 Data-Out buffer attribute

The data-out buffer attribute contains the Data-Out Buffer argument from an Execute Command procedure call (see 5.1).

4.5.13.10 Data-Out buffer size attribute

The data-out buffer size attribute contains the Data-Out Buffer Size argument from an Execute Command procedure call (see 5.1).

4.5.13.11 CRN attribute

The CRN attribute contains the CRN of the command (see 5.4.2.2).

4.5.13.12 Task priority attribute

The task priority attribute contains the priority of the command (see 8.7).

4.5.13.13 Sense data attribute

The ~~task management request~~ sense data attribute contains ~~a task management request (see clause 7)~~the sense data for the completed command (see 5.4.2.5).

4.5.13.14 Sense data length attribute

The sense data length attribute contains the length of the sense data for the completed command (see 5.4.2.5)

4.5.13.15 Retry delay timer attribute

The retry delay timer attribute contains the retry delay time for the completed command (see 5.4.2.5)

4.5.13.16 First burst enabled attribute

The first burst enabled attribute contains specifies if first burst for the command is enabled (see 5.4.2.2).

4.5.14 LogicaL Unit class

4.5.14.1 LogicaL Unit class overview

~~A~~ The Logical Unit class (see figure 5) contains the:

a) Device Server class (see 4.5.25);
b) Task Manager class (see 4.5.26);
c) Task Management Function class; and
d) Task Set class (see 4.5.27).

~~A~~ The Logical Unit class (see figure 5) may be substituted with the:

a) Well Known Logical Unit class (see 4.5.24.1); or
b) Hierarchical Logical Unit class.

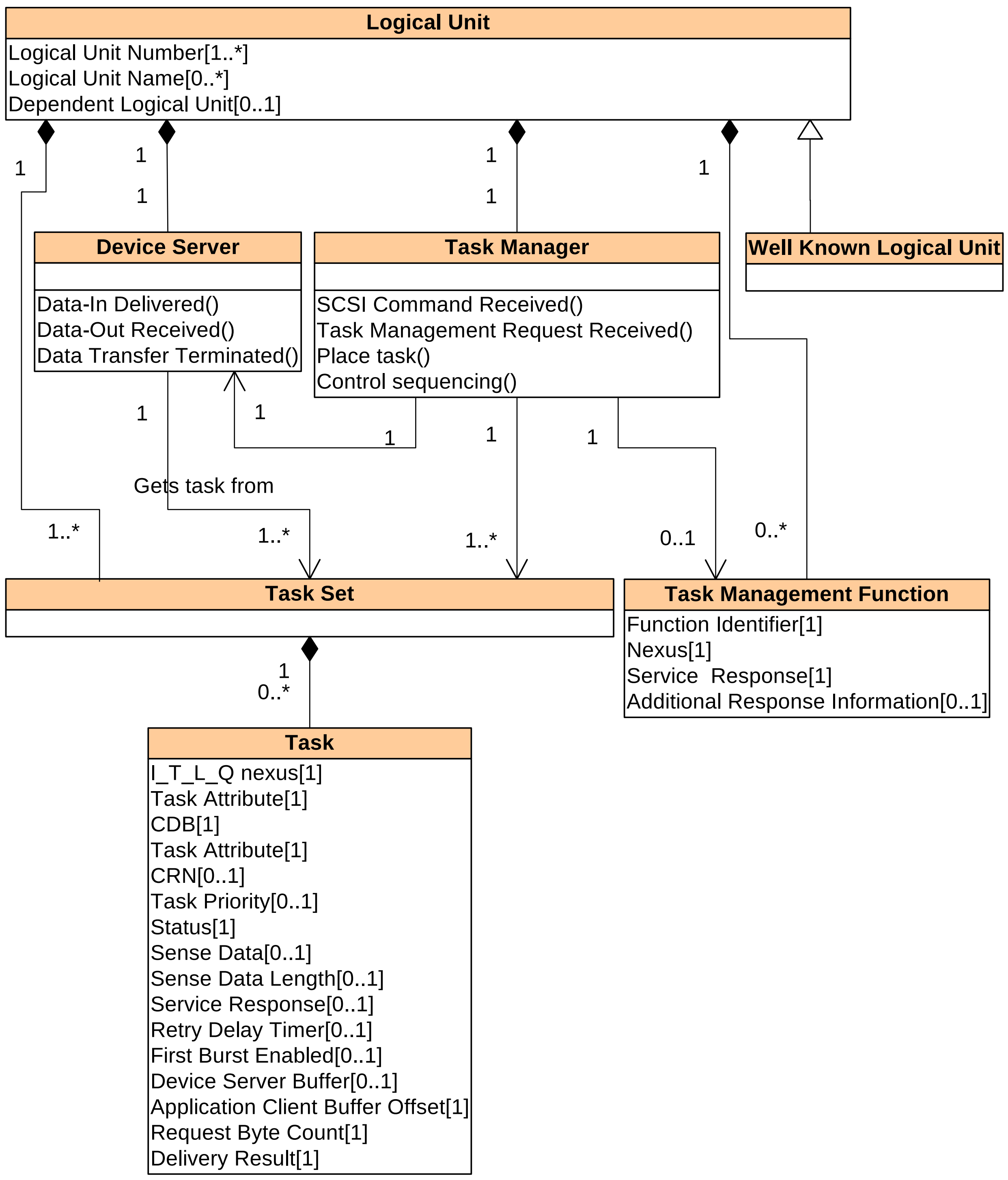

Figure 5 — Logical unit class diagram

Each instance of a logical unit class shall contain the following objects:

a) one device server;
b) one task manager;
c) zero or more task management functions; and
d) one or more task sets.

~~A~~ The logical unit class 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)~~(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.

4.5.14.2 Logical unit number attribute

~~A~~ The logical unit number attribute identifies the logical unit within a SCSI target device when accessed by a SCSI target port. If any logical unit within the scope of a SCSI target device includes one or more dependent logical units (see 4.5.24.4) 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~~5.8.4 and ~~7.12~~7.12.

4.5.14.3 Logical unit name attribute

~~A~~ The logical unit name attribute identifies a name ~~(see 3.1.67)~~(see 3.1.67) 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.

4.5.14.4 Dependent logical unit attribute

~~A~~ The 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:

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.

4.5.15 Device server class

~~The device server class processes commands~~.

The Device Server class processes the:

a) Data-In Delivered operation (see xx.xx.xx) to determine when data requested to be sent has been sent;
b) Data-Out operation (see xx.xx.xx) to determine when data requested to be received has been received; and
c) Data Transfer Terminated operation (see xx.xx.xx) to determines when a requested termination of a data transfer has been terminated; and
d) commands.

4.5.16 Task manager class

The Task Manager class processes the:

a) SCSI Command Received operation (see xx.xx.xx) to determine when a task has been received; ~~receive tasks from a task router~~;
b) Place Task operation to place tasks into a task set;
a) Control Sequencing operation to control~~s~~ the sequencing of one or more tasks within a logical unit;
b) Task Management Request Received operation (see xx.xx.xx) to determine when a task management function has been received; and
c) ~~processes the~~ task management functions ~~(see clause 7)~~(see clause 7).

4.5.17 Task set class

~~A~~ The task set class (see figure 5) contains:

a) a task class ~~(see 4.5.23)~~(see 4.5.14).

Each instance of a task set class shall contain the following objects:

a) zero or more tasks.

The interactions among the tasks in a task set are determined by the requirements for task set management specified in ~~clause 8~~clause 8 and the ACA requirements specified in ~~5.8.1~~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).

4.5.18 Task class

4.5.18.1 Task class overview

~~A~~ The 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~~ The 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).

~~4.5.18.2 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.~~

4.5.18.3 I_T_L_Q nexus attribute

The I_T_L_Q nexus attribute contains the I_T_L_Q nexus of the task (see 4.7).

4.5.18.4 CDB attribute

The CDB attribute contains a CDB (see 5.2 and SPC-3) that defines the work to be performed by a logical unit.

4.5.18.5 ~~CDB~~ Task Attribute attribute

A Task Attribute attribute (see 8.6) contains the task attribute (e.g., simple, ordered, head or queue, ACA) of ~~identifies~~ a command.

4.5.18.6 CRN attribute

The CRN attribute contains the CRN of the command (see 5.4.2.2).

4.5.18.7 Task priority attribute

The task priority attribute contains the priority of the command (see 8.7).

4.5.18.8 Status attribute

The status attribute contains the status of the completed command (see 5.3).

4.5.18.9 Service response attribute

The service response attribute contains the service response for the completed command (see 5.4.2.5)

4.5.18.10 Sense data attribute

The ~~task management request~~ sense data attribute contains ~~a task management request (see clause 7)~~the sense data for the completed command (see 5.4.2.5).

4.5.18.11 Sense data length attribute

The sense data length attribute contains the length of the sense data for the completed command (see 5.4.2.5)

4.5.18.12 Service Response attribute

The Service Response attribute contains the service response (see 5x.1x).

4.5.18.13 Retry delay timer attribute

The retry delay timer attribute contains the retry delay time for the completed command (see 5.4.2.5)

4.5.18.14 First burst enabled attribute

The first burst enabled attribute contains specifies if first burst for the command is enabled (see 5.4.2.2).

4.5.18.15 Device Server Buffer attribute

The Device Server Buffer attribute contains the Device Server Buffer argument used for Send Data-In procedure calls (see x.x) and a Receive Data-Out procedure calls (see x.x).

4.5.18.16 Application Client Buffer Offset attribute

The Application Client Buffer Offset attribute contains the Application Client Buffer Offset argument used for Send Data-In procedure calls (see x.x) and a Receive Data-Out procedure calls (see x.x).

4.5.18.17 Request Byte Count attribute

The Request Byte Count attribute contains the Request Byte Count argument used for Send Data-In procedure calls (see x.x) and a Receive Data-Out procedure calls (see x.x).

4.5.18.18 Delivery Result attribute

The Delivery Result attribute contains the Delivery Result argument from a Data-In Delivered procedure call (see x.x) or a Data-Out Received procedure call (see x.x).

4.5.19 Task Management Function class

4.5.19.1 Task ~~class~~ Management Function class overview

~~A~~ The Task Management Function class ~~is an application client class that~~ represents a ~~command (see clause 5)~~SCSI task management function (see clause 7).

4.5.19.2 Nexus attribute

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

4.5.19.3 Function Identifier attribute

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

4.5.19.4 Service Response attribute

The Service Response attribute contains the service response (see 5x.1x).

4.5.19.5 Additional response information attribute

The additional response information attribute contains any additional response information for the task management function (see clause 7).

4.5.20 Well known logical unit class

~~A~~ The well known logical unit class is a logical unit class (see 4.5.24.1) 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~~5.8.4 and ~~7.12~~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 4.5.19).

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

The nexus object represents a relationship between 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).

Table 1 maps the nexus object to other identifier objects.

Table 1 — Mapping nexus to SAM-2 identifiers

Nexus	Identifiers contained in nexus	Reference
I_T	Initiator port identifier Target port identifier	4.5.9 4.5.14
I_T_L	Initiator port identifier Target port identifier Logical unit number	4.5.9 4.5.14 4.5.19
I_T_L_Q	Initiator port identifier Target port identifier Logical unit number Task identifier ~~tag~~	4.5.9 4.5.14 4.5.19 4.5.18.2

5.4 SCSI transport protocol services in support of Execute Command

5.4.1 Overview

5.4.2 Command and Status SCSI transport protocol services

5.4.2.1 Command and Status SCSI transport protocol services overview

5.4.2.2 Send SCSI Command transport protocol service request

5.4.2.3 SCSI Command Received transport protocol service indication

~~A~~ The task router (see x.x.x) ~~SCSI target port~~ uses the SCSI Command Received transport protocol service indication to notify a task manager ~~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 4.7).
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.

7.1 Introduction

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:

Service Response = Function name (IN (nexus), OUT ([additional response information])

The task management function names are summarized in table 2.

Table 2 — Task Management Functions

Task Management Function	Nexus	Additional Response Information argument supported	Reference
ABORT TASK	I_T_L_Q	no	7.2
ABORT TASK SET	I_T_L	no	7.3
CLEAR ACA	I_T_L	no	7.4
CLEAR TASK SET	I_T_L	no	7.5
I_T NEXUS RESET	I_T	no	7.6
LOGICAL UNIT RESET	I_T_L	no	7.7
QUERY TASK	I_T_L_Q	no	7.8
QUERY TASK SET	I_T_L	no	7.9
QUERY UNIT ATTENTION	I_T_L	yes	7.10

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 identifier ~~tag~~ nexus (see 4.7).

Output arguments:

Additional Response Information: If supported by the SCSI 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). SCSI transport protocols may or may not support the Additional Response Information argument. A SCSI 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 service responses shall be returned:

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: A 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, QUERY UNIT ATTENTION, or QUERY UNIT ATTENTION).

FUNCTION REJECTED: A 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: A 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 3.1.111) or SCSI target device malfunction. The task manager may or may not have successfully performed the specified function.

Each SCSI transport protocol standard shall define the events for 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, QUERY TASK, QUERY TASK SET, or QUERY UNIT ATTENTION 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.

7.12 Task management SCSI transport protocol services

7.12.1 Task management SCSI transport protocol services overview

7.12.2 Send Task Management Request transport protocol service request

7.12.3 Task Management Request Received transport protocol service indication

A task router (see x.x.x) ~~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 function.

Editor's Note 1: The above change needs to also be made in SAS-2.

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 4.7).
Function Identifier:	Argument encoding the task management function to be performed.

7.12.4 Task Management Function Executed transport protocol service response

A task manger uses the 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))

Input arguments:

Nexus:	An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.7).
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 3.1.111) or SCSI target device malfunction. The task manager may or may not have successfully performed the specified function.

Annex A
(informative)

Terminology mapping

The introduction of a UML model into this standard resulted in changes in terminology between SAM-4 and SAM (see table A.1).

Table A.1 — SAM-4 to SAM terminology mapping

SAM-4 equivalent term	SAM term
task identifer	task tag