Date: May 22, 2006

To: T10 Committee (SCSI)

From: George Penokie (IBM/Tivoli)

Subject: SAM-4: Converting to UML part 1

1 Overview

The current SCSI architecture follows no particular documentation convention outside of it's own T10 invented one. This proposal suggests that the documenting convention be changed to comply with the unified modeling language (UML) standards. UML is widely used and, as such, would make the concepts defined in SAM-4 more understandable to a larger audience.

For those interested in learning about UML I would suggest the following reading materials:

a) SAMS Teach Yourself UML in 24 hours (This is a kind of crash course that will give you the basics);
b) The Unified Modeling Language User Guide by Grady Booch, et al. (This is considered the UML bible by many);
c) The Unified Modeling Language Reference Manual by James Rumbaugh, et al. (This is considered the UML bible part 2 by many); and
d) UML Distilled Second Edition A brief Guide to the Standard Object Modeling Language by Martin Fowler (This is a good reference book).

There are many more and there is no lack of opinion as to which is most useful. Also, if you are a real masochist you can read the standard. Go to http://www.omg.org/cgi-bin/apps/doclist.pl web site and search on UML. What you are looking for is the Unified Modeling Language: OCL version 2.0 ptc/03-08-08.

2 The first step - class vs. object

The first thing that needs to be done to accomplish this conversion is to understand that most of the things SAM-4 currently calls objects are not objects but classes.

Classes are defined in UML as:

2.0.1 class: A description of a set of objects that share the same attributes, operations, relationships, and semantics. Classes are used to represent software things, hardware things, and things that are purely conceptual. Every class has a class name. Classes may have attributes and may support operations.

Objects are defined in UML as:

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

Some supporting UML terminology:

2.0.3 attributes: A named property of a class that describes the range of values that the class or its instances (i.e., objects) may hold.

2.0.4 behavior: How an object acts and reacts to requests from other objects.

2.0.5 class name: A able of a class.

2.0.6 constraint: A mechanism for specifying semantics or conditions that are maintained as true between entities (e.g., a required condition between associations).

2.0.7 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.

2.0.8 state: One of the possible conditions in which an object may exist that normally changes over time and is represented by attributes.

For a non-SCSI example of classes and objects see figure 1. For a SCSI example of classes and objects see figure 2.

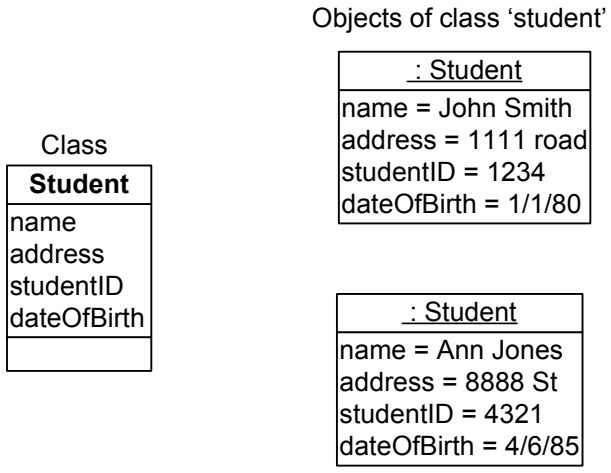


Figure 1 — NonSCSI example of class and object representation

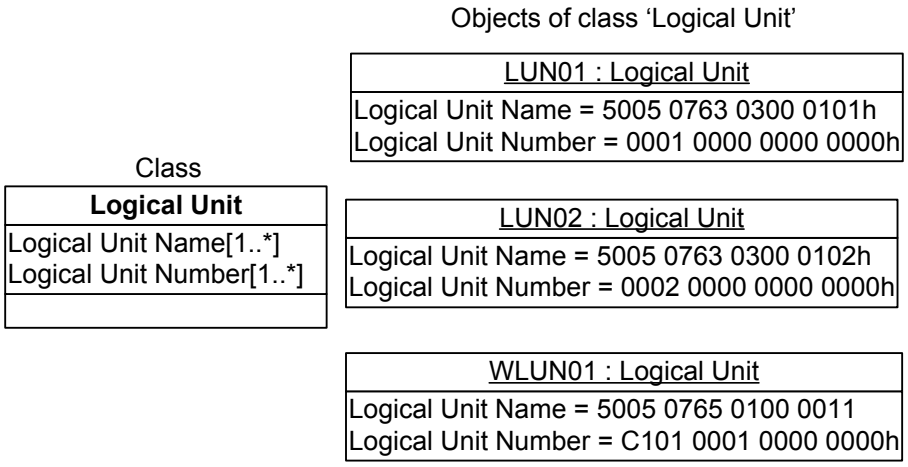


Figure 2 — SCSI example of class and object representation

3 Example of a class diagram

For an example of what figures in SAM-4 would look like a converted logical unit model (see figure 3) as currently shown in SAM-4 would look like what is shown in figure 4.

.

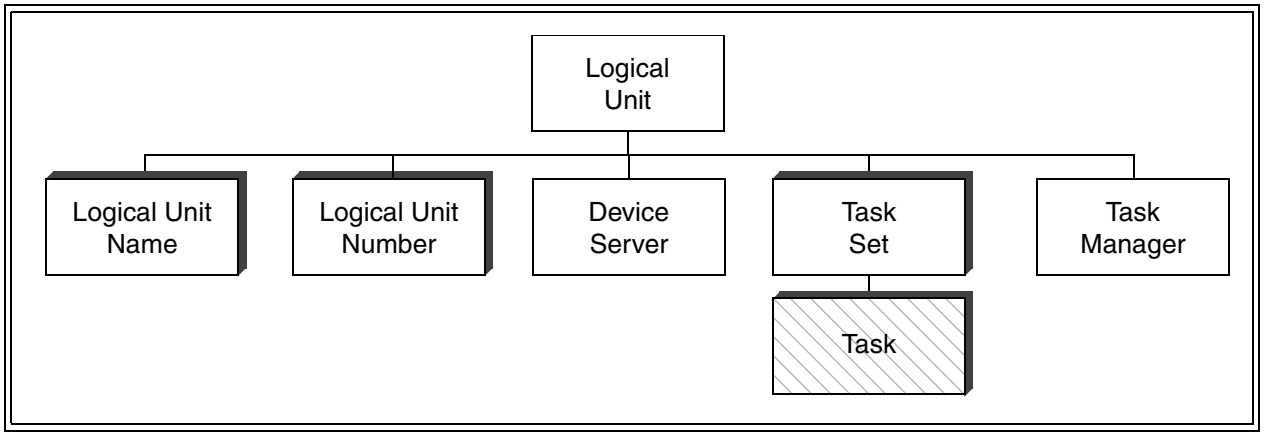


Figure 3 — Logical unit model

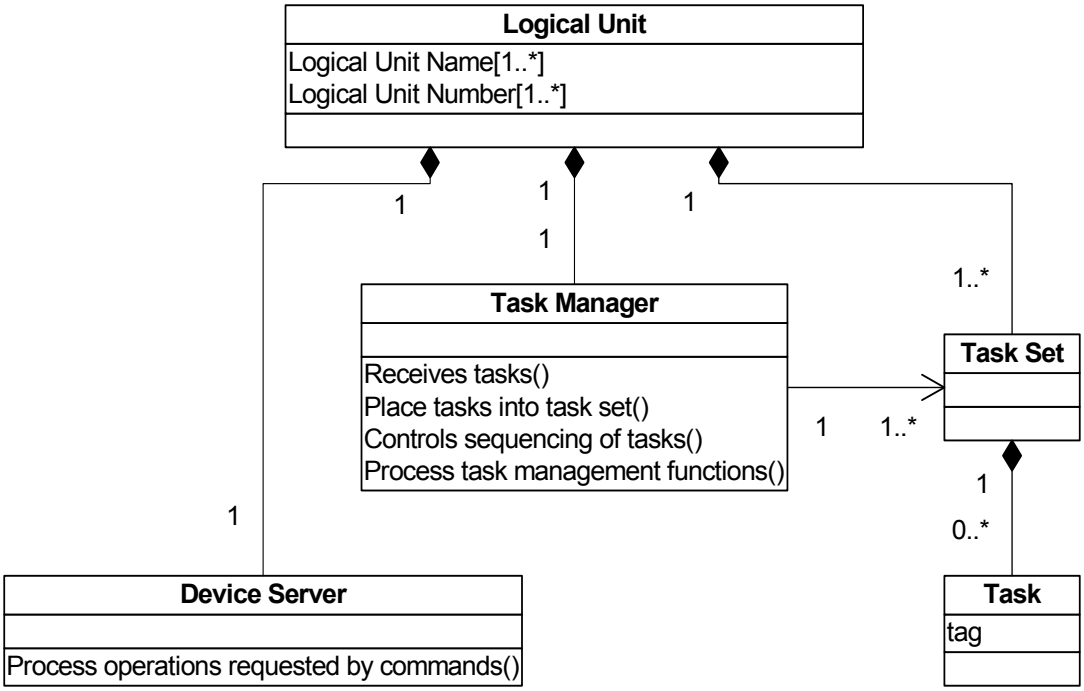


Figure 4 — Logical unit class diagram

1 SAM-4 changes

2 Normative references

3 Definitions, symbols, abbreviations, and conventions

3.1 Definitions

3.1.1 aggregation: When used in class diagrams, a form of association that defines a whole-part relationship between the whole (i.e., aggregate) and its parts.

3.1.2 association: When used in class diagrams, 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).

3.1.3 attribute: When used in class diagrams, a named property of a class that describes the range of values that the class or its objects may hold. When used in object diagrams, a named property of the object.

3.1.4 constraint: When used in class diagrams and object diagrams, a mechanism for specifying semantics or conditions that are maintained as true between entities (e.g., a required condition between associations).

3.1.5 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.

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

3.1.7 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).

3.1.8 generalization: When used in class diagrams, a relationship among classes where one class (i.e., super-class) shares the attributes and operations on one or more classes (i.e., subclasses).

3.1.9 multiplicity: When used in class diagrams, an indication of the range of allowable instances that a class or an attribute may have.

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

3.1.11 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.

3.1.12 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.

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

3.1.14 role: When used in class diagrams and object diagrams, 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.

3.2 Editorial conventions

3.3 Numeric conventions

3.4 Notation conventions

3.4.1 Class diagram conventions

The notation in this subclause is based on the Unified Modeling Language (UML) specification.

Figure 1 shows the notation used for classes in class diagrams.

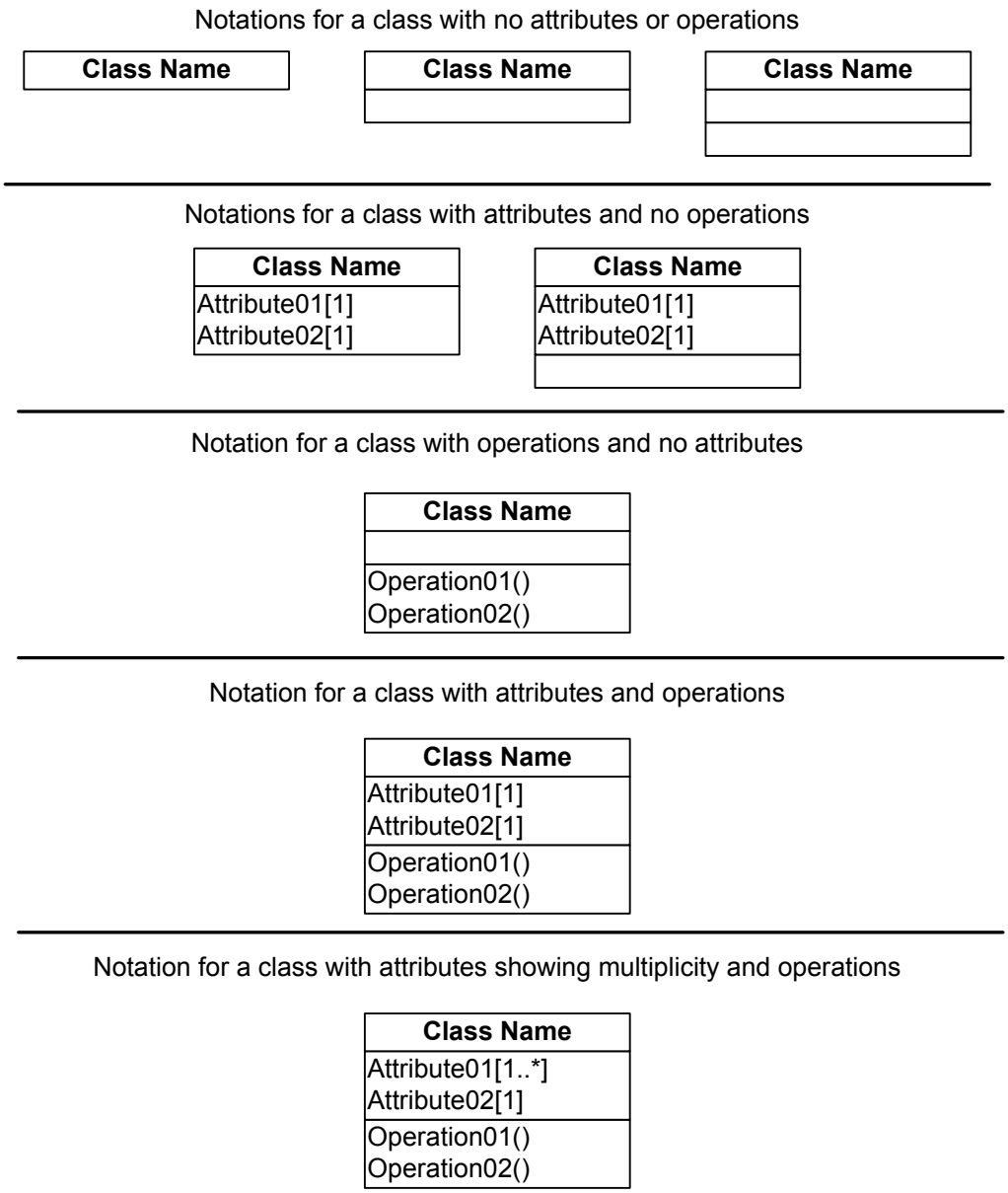


Figure 1 — Class diagram conventions

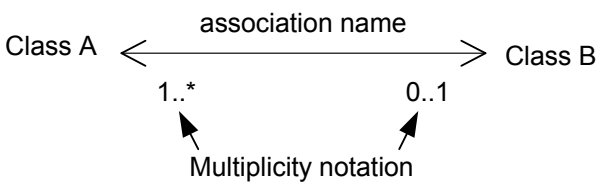
See table 1 for the notation used to indicate multiplicity.

Table 1 — Multiplicity notation

Notation	Description
not specified	The number of instances of an attribute is not specified.
1	One instance of the class or attribute exists.
0..*	Zero or more instances of the class or attribute exist.
1..*	One or more instances of the class or attribute exist.
0..1	Zero or one instance of the class or attribute exists.
n..m	n to m instances of the class or attribute exist (e.g., 2..8).
x, n..m	Multiple disjoint instances of the class or attribute exist (e.g., 2, 8..15).
[a] See figure 2 and figure 3 for examples of multiplicity notation.	

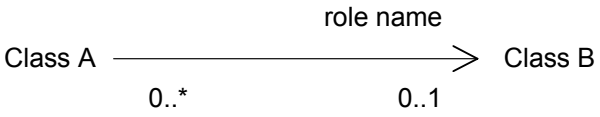
Solid lines with arrowheads (see figure 2) are the notation used to describe the association relationship between classes in class diagrams. Multiplicity notation occurs at each end of the solid line.

Association (“knows about” relationship)



Class A knows about Class B (i.e., read as “Class A association name Class B”) and Class B knows about Class A (i.e., read as “Class B association name Class A”)

Class B knows about Class A (i.e., read as “Class B knows about Class A”) but Class A does not know about Class B



Class A knows about Class B (i.e., read as “Class A uses the role name attribute of Class B”) but Class B does not know about Class A

Note: The role name and association name are optional

Examples of class diagrams using associations

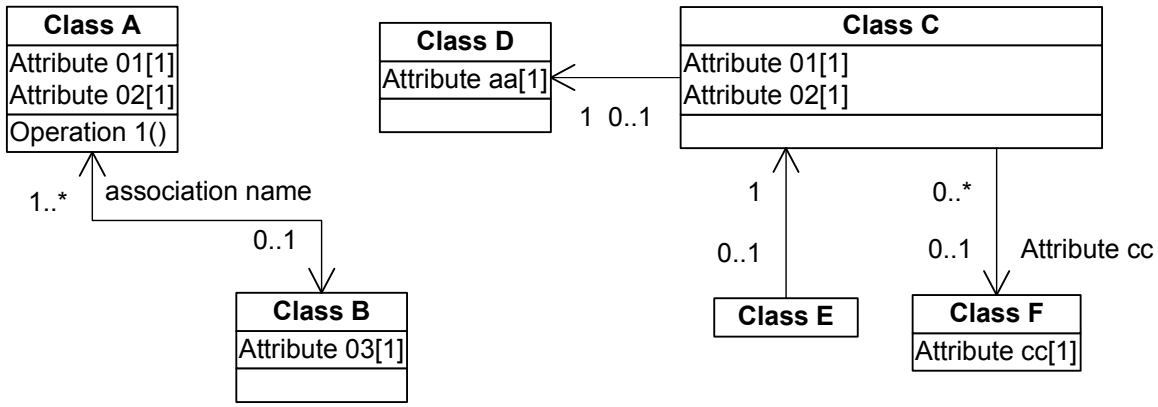
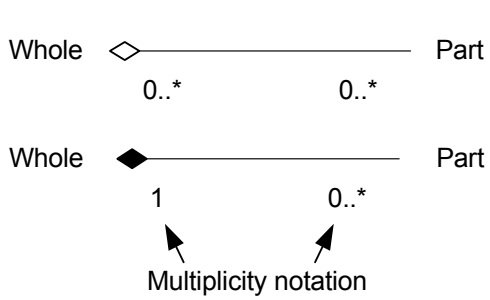


Figure 2 — Notation for association relationships for class diagrams

Solid lines with diamonds (see figure 3) are the notation used to describe the aggregation relationship between classes in class diagrams. Multiplicity notation occurs at each end of the solid line.

Aggregation (“is a part of” or “contains” relationship)



The part class is part of the whole class and may continue to exist even if the whole class is removed. Read as “the whole contains the part.”

The part class is part of the whole class, shall only belong to one whole class, and shall not continue to exist if the whole class is removed. Read as “the whole contains the part.”

Examples of class diagrams using aggregation

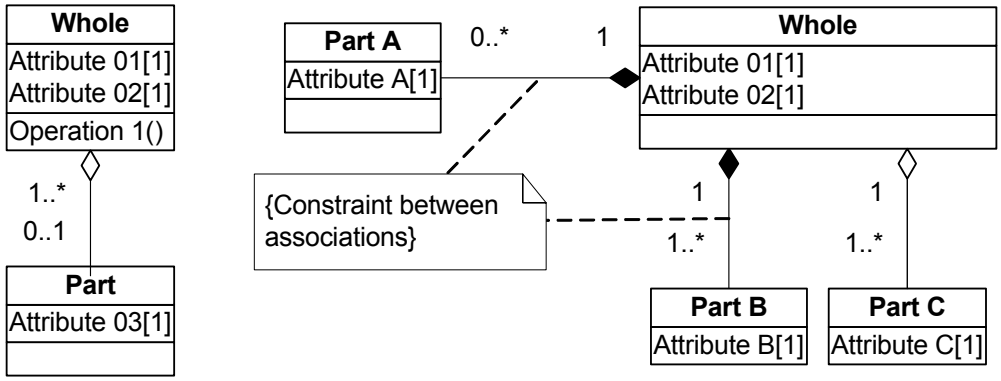


Figure 3 — Notation for aggregation relationships for class diagrams

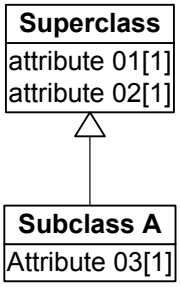
Solid lines with triangles (see figure 4) are the notation used to describe the generalization relationship between classes in class diagrams.

Generalization (“is a kind of” relationship)

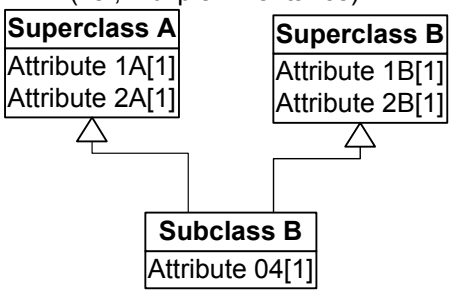
Subclass is a kind of superclass. A subclass shares all the attributes and operations of the superclass (i.e., the subclass inherits from the superclass).

Examples of class diagrams using generalization

Single superclass/single subclass:



Multiple superclass/single subclass (i.e., muliple inheritance):



Single superclass/multiple subclass:

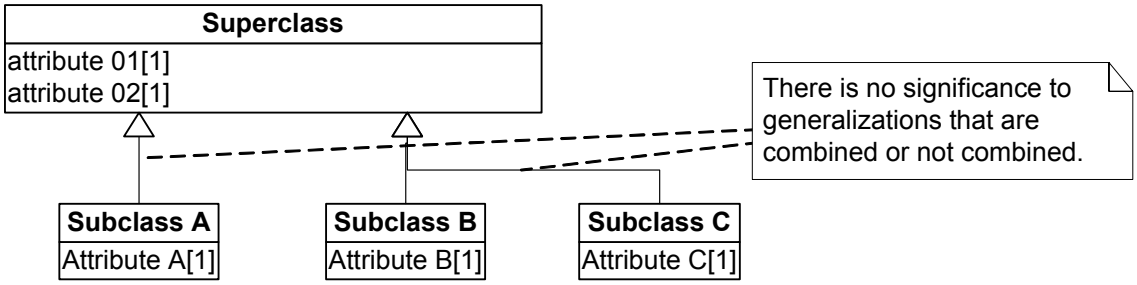
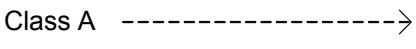


Figure 4 — Notation for generalization relationships for class diagrams

Dashed lines with arrowheads (see figure 5) are the notation used to describe the dependency relationship between classes in class diagrams.

Dependency (“depends on” relationship)



Class A depends on class B. A change in class B may cause a change in class A.

Example of class diagram using dependency

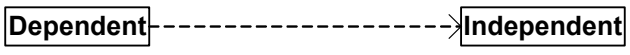


Figure 5 — Notation for dependency relationships for class diagrams

3.4.2 Object diagram conventions

Figure 6 shows the notation used for objects in object diagrams.

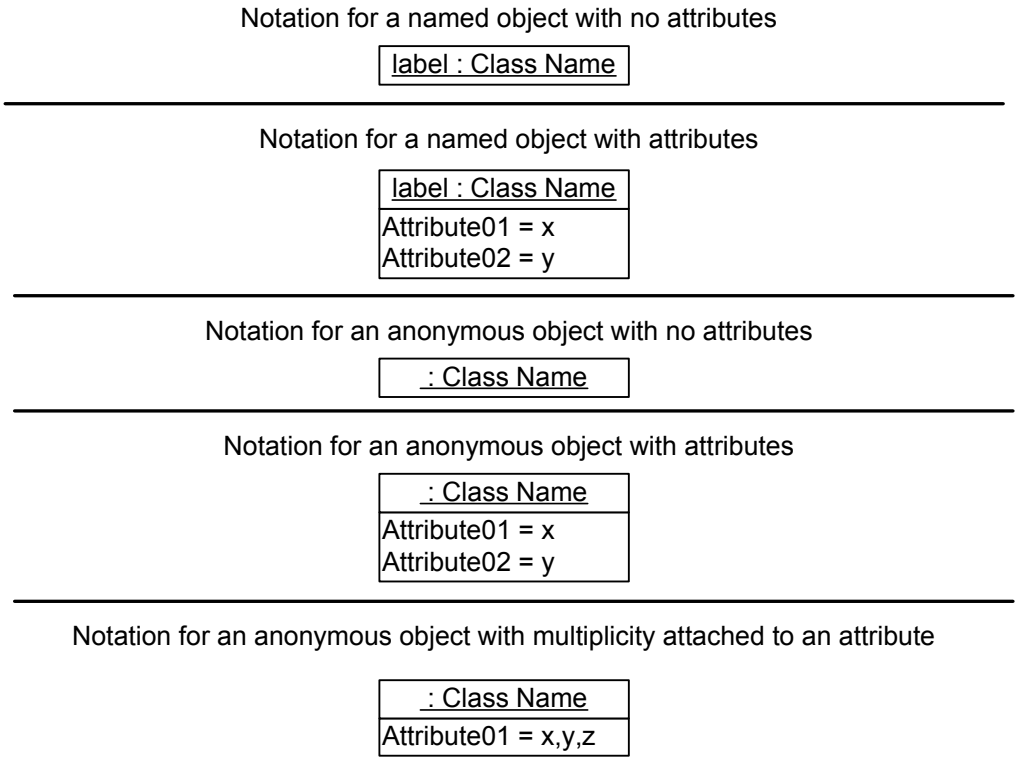


Figure 6 — Object diagram conventions

Solid lines (see figure 4) are the notation used to describe the link relationship between objects in object diagrams.

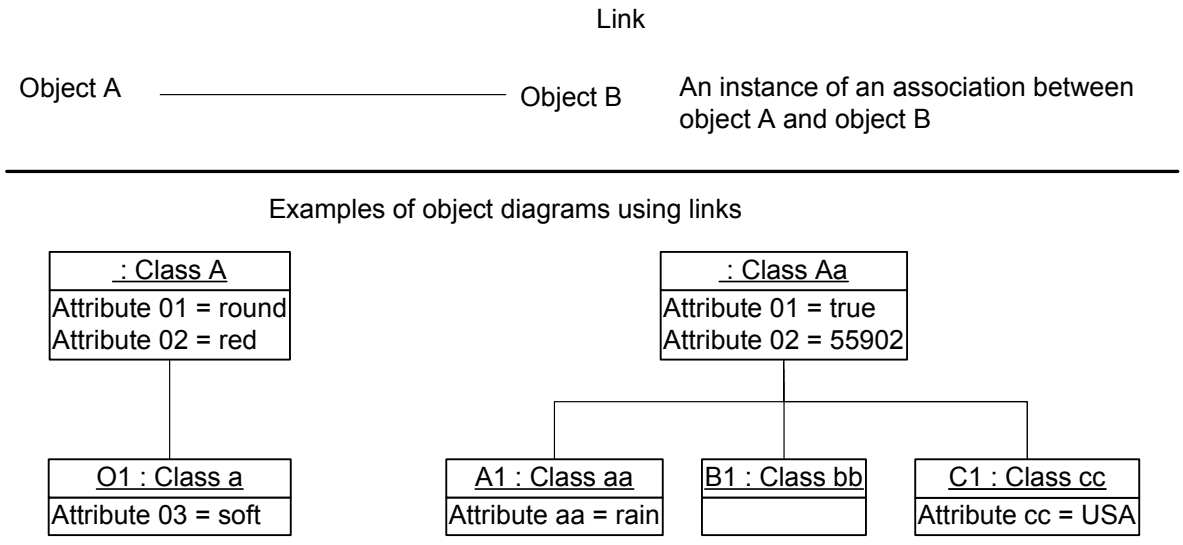


Figure 7 — Notation for link relationships for object diagrams

3.4.3 Notation for procedure calls

3.4.4 Notation for state diagrams

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 3.1.5), 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 a complex entity that performs 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 7.10. The model describes required behavior in terms of layers, classes within layers and SCSI transport protocol service transactions between layers.

4.2 The SCSI distributed service model

Service interfaces between distributed classes are represented by the client-server model shown in figure 8. Dashed horizontal lines with arrowheads denote a single request-response transaction as it appears to the client and server. The solid lines with arrowheads indicate the actual transaction path through the service delivery subsystem. In such a model, each client or server is a single thread of processing that runs concurrently with all other clients or servers.

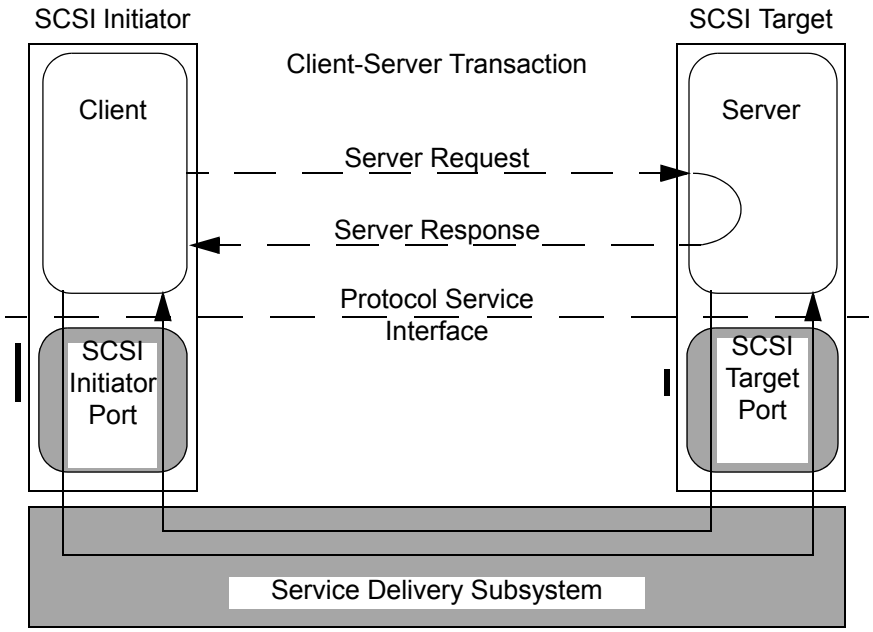


Figure 8 — Client-Server model

A client-server transaction is represented as a procedure call with inputs supplied by the caller (i.e., the client). The procedure call is processed by the server and returns outputs and a procedure call status. A client directs requests to a remote server via the SCSI initiator port and service delivery subsystem and receives a completion response or a failure notification. The request identifies the server and the service to be performed and includes the input data. The response conveys the output data and request status. A failure notification indicates that a condition has been detected (e.g., a reset or service delivery failure) that precludes request completion.

As seen by the client, a request becomes pending when it is passed to the SCSI initiator port for transmission. The request is complete when the server response is received or when a failure notification is sent. As seen by the server, the request becomes pending upon receipt and completes when the response is passed to the SCSI target port for return to the client. As a result there may be a time skew between the server and client's perception of request status and server state. All references to a pending command or task management function in this standard are from the application client's point of view (see 5.5 and 7.9).

Client-server relationships are not symmetrical. A client may only originate requests for service. A server may only respond to such requests.

The client requests an operation provided by a server located in another SCSI device and waits for completion, which includes transmission of the request to and response from the remote server. From the client's point of view, the behavior of a service requested from another SCSI device is indistinguishable from a request processed in the same SCSI device. In this model, confirmation of successful request or response delivery by the sender is not required. The model assumes that delivery failures are detected by the SCSI initiator port or within the service delivery subsystem.

4.3 The SCSI client-server model

4.3.1 SCSI client-server model overview

As shown in figure 9, 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 or series of linked commands 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.

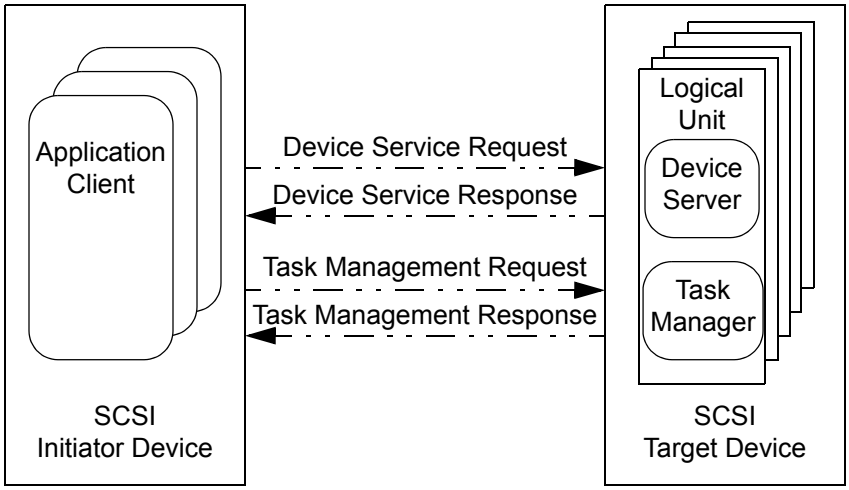


Figure 9 — SCSI client-server model

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 single command, a series of linked commands, 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, series of linked commands, 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. If supported by a logical unit, a series of linked commands may be used to define an extended I/O operation.

4.3.2 Synchronizing client and server states

One way a client is informed of changes in server state is through the arrival of server responses. Such state changes occur after the server has sent the associated response and possibly before the response has been received by the SCSI initiator device (e.g., the SCSI target device changes state upon processing the **Send Command Complete** procedure call (see 5.4.2), but the SCSI initiator device is not informed of the state change until the **Command Complete Received** SCSI transport protocol service confirmation arrives).

SCSI transport protocols may require the SCSI target device to verify that the response has been received successfully before completing a state change. State changes controlled in this manner are said to be synchronized. Since synchronized state changes are not assumed or required by the architecture model, there may be a time lag between the occurrence of a state change within the SCSI target device and the SCSI initiator device's awareness of that change.

This standard assumes that state synchronization, if required by a SCSI transport protocol standard, is enforced by the service delivery subsystem transparently to the server (i.e., whenever the server invokes a SCSI transport protocol service to return a response as described in 7.10 and 5.4. It is assumed that the SCSI port for such a SCSI transport protocol does not return control to the server until the response has been successfully delivered to the SCSI initiator device).

4.3.3 Request/Response ordering

Request or response transactions are said to be in order if, relative to a given pair of sending and receiving SCSI ports, transactions are delivered in the order they were sent.

A sender may require control over the order in which its requests or responses are presented to the receiver (e.g., the sequence in which requests are received is often important whenever a SCSI initiator device issues a series of commands with the ORDERED attribute to a logical unit as described in clause 8). In this case, the order in which these commands are completed, and hence the final state of the logical unit, may depend on the order in which these commands are received. The SCSI initiator device may develop knowledge about the state of pending commands and task management functions and may take action based on the nature and sequence of SCSI target device responses (e.g., a SCSI initiator device should be aware that further responses are possible from an aborted command because the command completion response may be delivered out of order with respect to the abort response).

The manner in which ordering constraints are established is vendor specific. An implementation may delegate this responsibility to the application client (e.g., the device driver). In-order delivery may be an intrinsic property of the service delivery subsystem or a requirement established by the SCSI transport protocol standard.

The order in which task management requests are processed is not specified by the SCSI architecture model. The SCSI architecture model does not require in-order delivery of such requests or processing by the task manager in the order received. To guarantee the processing order of task management requests referencing a specific logical unit, an application client should not have more than one such request pending to that logical unit.

To simplify the description of behavior, the SCSI architecture model assumes in-order delivery of requests or responses to be a property of the service delivery subsystem. This assumption does not constitute a requirement. The SCSI architecture model makes no assumption about and places no requirement on the ordering of requests or responses for different I_T nexuses.

4.4 The SCSI structural model

The SCSI structural model represents a view of the classes comprising a SCSI I/O system as seen by the application clients interacting with the system. As shown in figure 10, the fundamental class is the SCSI domain that represents an I/O system. A SCSI domain is made up of SCSI devices and a service delivery subsystem that transports commands, data, task management functions, and related information. A SCSI device contains clients or servers or both and the infrastructure to support them.

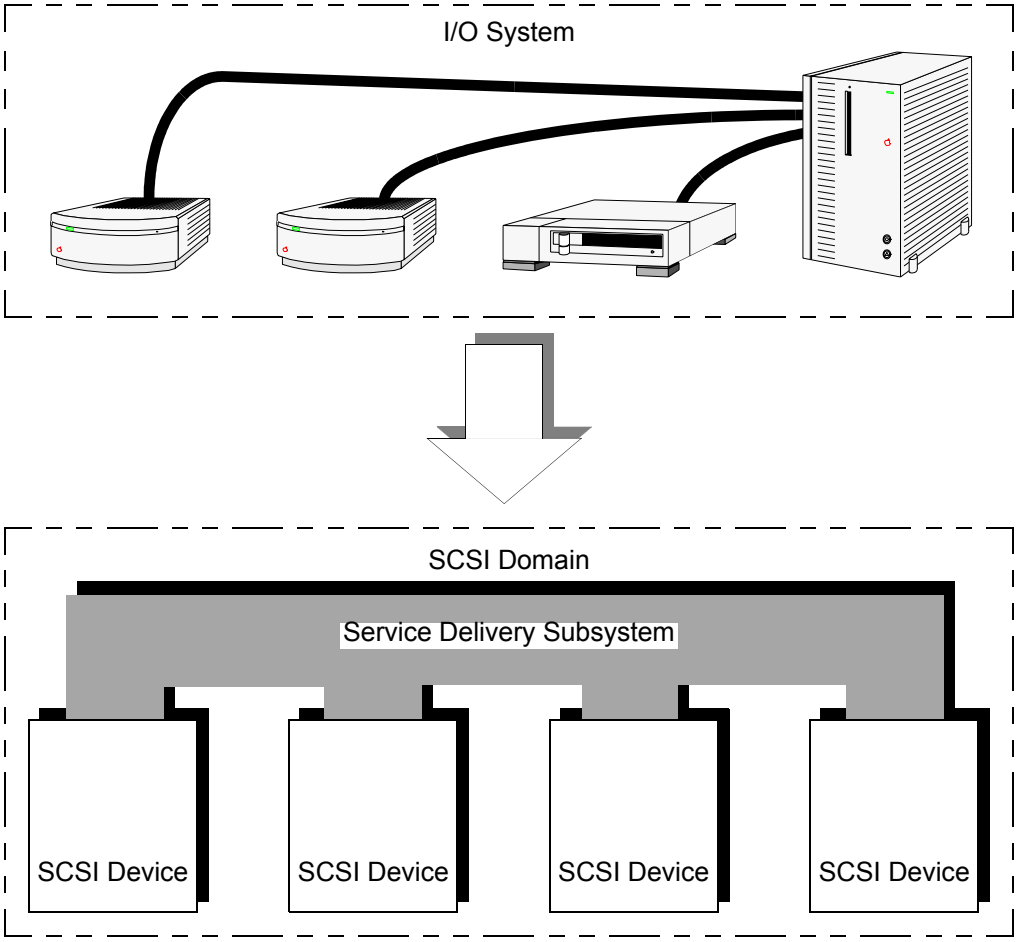


Figure 10 — SCSI I/O system and domain model

4.5 SCSI classes

4.5.1 SCSI classes overview

Figure 11 shows the main functional classes of the SCSI domain. This standard defines these classes in greater detail.

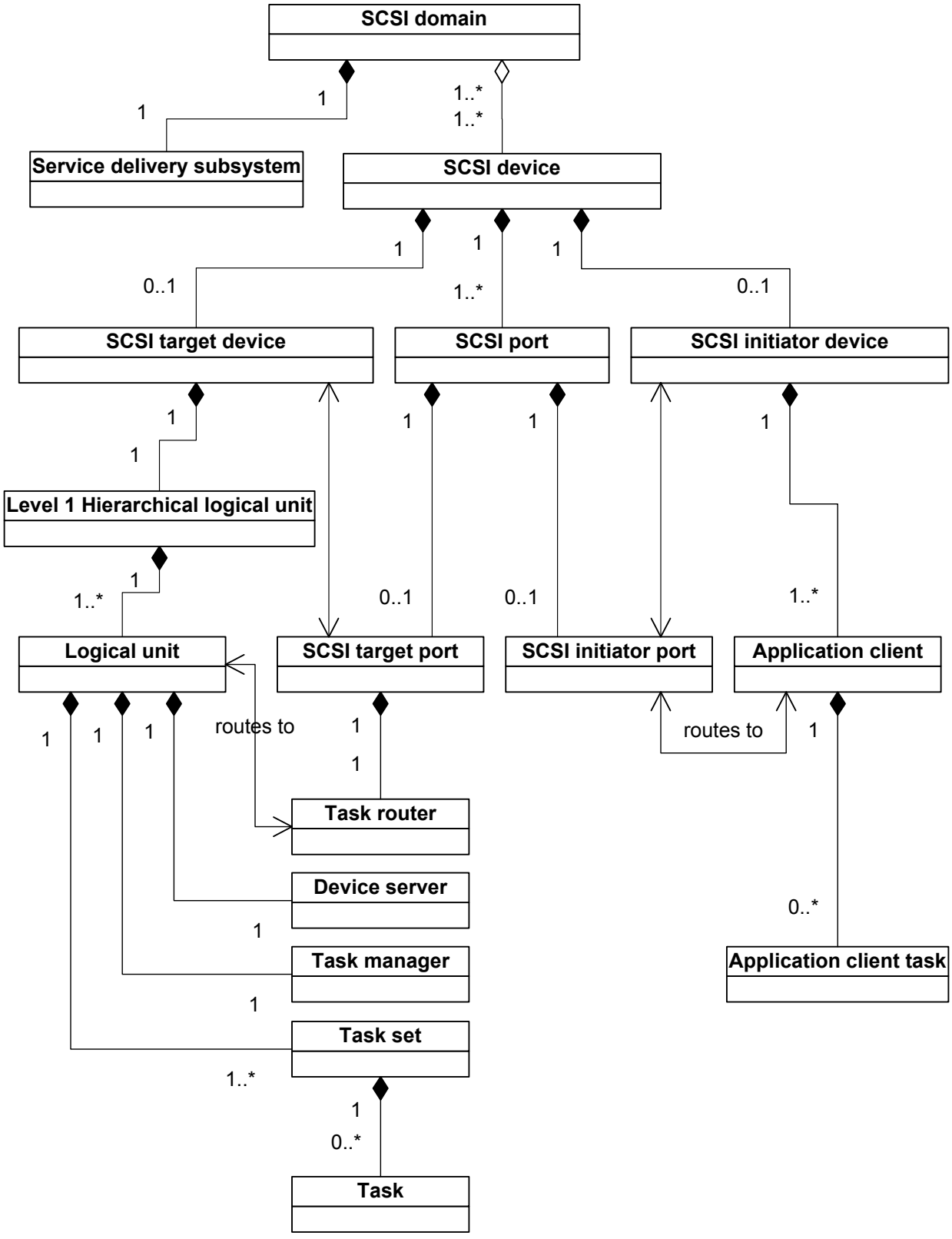


Figure 11 — SCSI domain class diagram overview

4.5.2 SCSI domain class

A SCSI domain class (figure 12) 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.

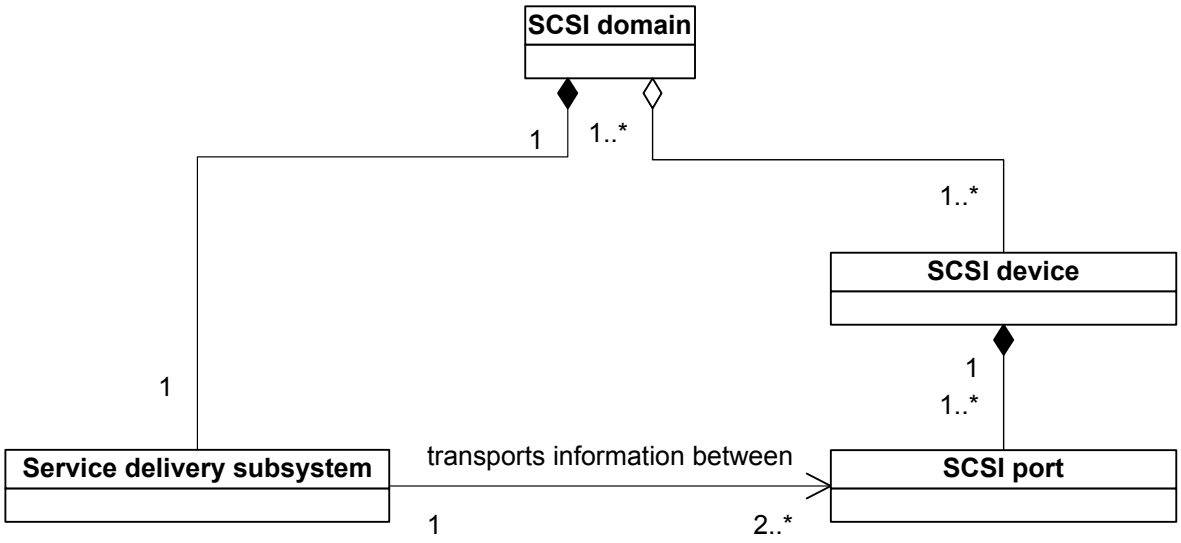


Figure 12 — SCSI domain class diagram

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.

See figure 13 for the instantiation of the minimum set of objects that make up a valid SCSI domain.

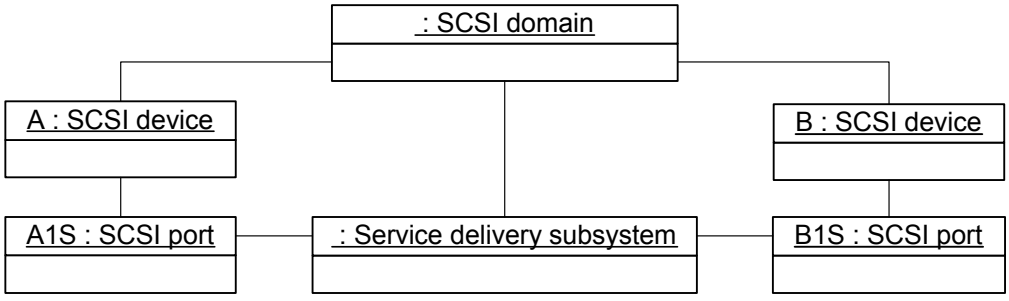


Figure 13 — SCSI domain object diagram

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 3.1.93) 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.

The 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 14 for the SCSI device class diagram.

A SCSI device class 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 4.5.15), or both.

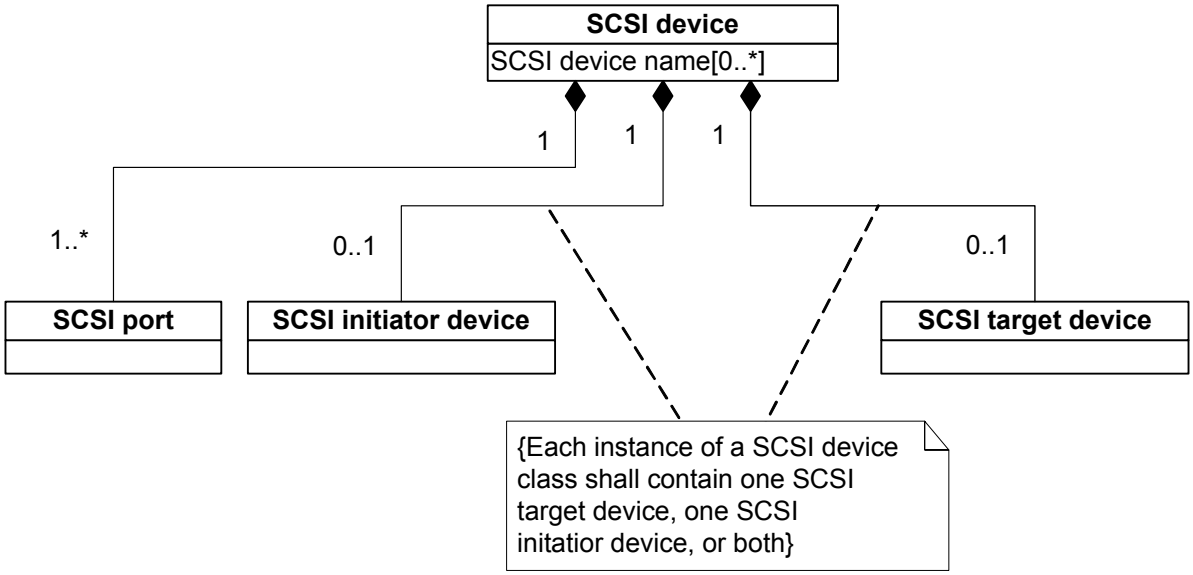


Figure 14 — SCSI device class diagram

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.

4.5.4.2 SCSI device name attribute

A SCSI device name attribute contains a name (see 3.1.63) 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

A SCSI port class (see figure 15) 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.

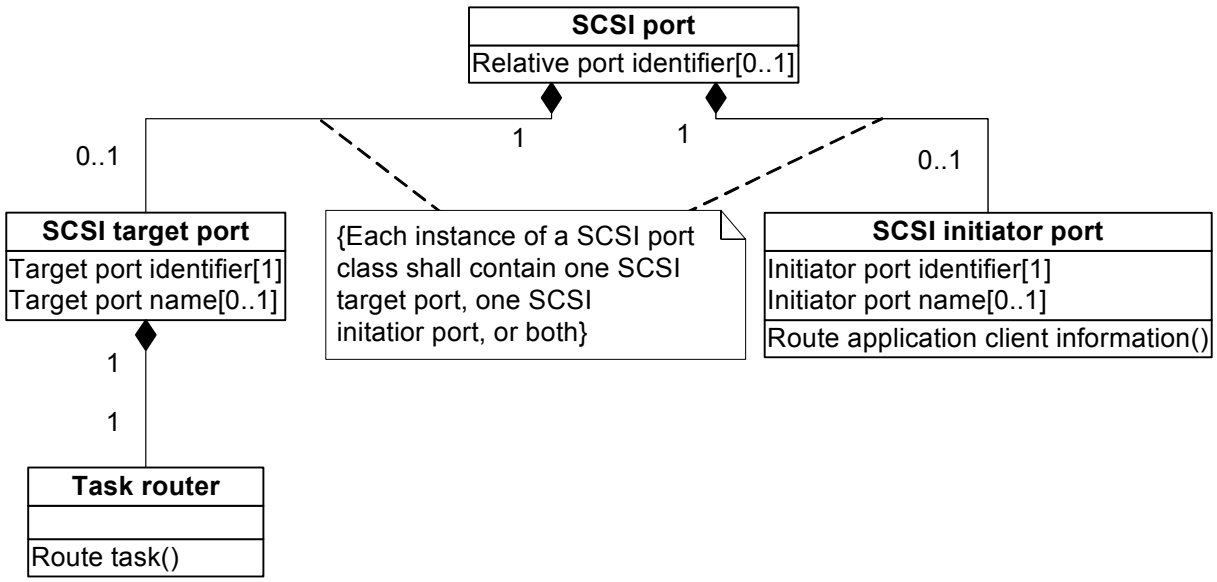


Figure 15 — SCSI port class diagram

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.

4.5.5.2 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.

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.

A SCSI port name attribute contains an optional name (see 3.1.63) of a SCSI port that is world wide unique within the SCSI transport protocol of the SCSI domain of that SCSI port. A SCSI port may have at most one name. A SCSI port name shall never change and may be used to persistently identify a SCSI initiator port or SCSI target port.

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

4.5.6.1 SCSI target port class overview

A SCSI target port class (see figure 15) contains the:

a) task router class (see 4.5.8);

The SCSI target port class connects SCSI target devices to the service delivery subsystem.

4.5.6.2 Target port identifier attribute

The target port identifier attribute contains a target port identifier (see 3.1.xxx) for a SCSI target port. The target port identifier is a value by which a SCSI target port is referenced within a domain.

4.5.6.3 Target port name attribute

A target port name attribute contains an optional name (see 3.1.63) 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.

4.5.7 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 the service delivery subsystem.

4.5.7.2 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.

4.5.7.3 Initiator port name attribute

A initiator port name attribute contains an optional name (see 3.1.63) 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.

4.5.8 SCSI task router class

The SCSI task router class routes information (e.g., commands and task management functions) between a logical unit and the 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, and LOGICAL UNIT RESET) 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 7.10.

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 16) 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).

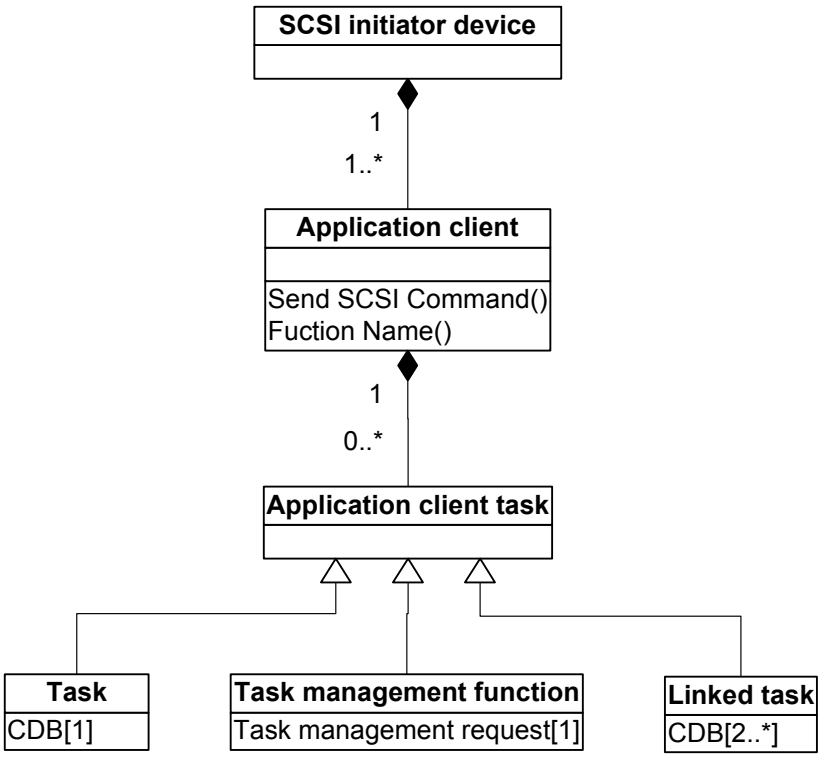


Figure 16 — SCSI initiator device class diagram

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 tasks.

4.5.10 Application client class

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

An application client class originates commands by issuing a **Send SCSI Command** requests (see xxx).

An application client class originates task management requests by issuing a Function name service request (see xxx).

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 7.11.

4.5.11 Application client task class

An application client task class (see figure 16) shall be substituted with:

a) a task class (see 4.5.12);
b) a task management function class (see 4.5.13); or
c) a linked task class (see 4.5.14).

An application client task class is the source for a single command, series of linked commands, or a single task management function.

4.5.12 Task class

4.5.12.1 Task class overview

A task class is an application client class that represents a task that consists of a single command (see clause 5x).

Each instance of a task class represents the work associated with a command. A new command causes the creation of a task. The 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 5.7.1.

4.5.12.2 CDB attribute

For a description of the CDB attribute contains a CDB as defined in 5.2x and SPC-3.

4.5.13 Task management function class

4.5.13.1 Task management function overview

A task management function class is an application client class that represents a SCSI task management function (see clause 7x).

4.5.13.2 Task management request attribute

For a description of the task management request attribute contains a task management request as defined in clause 7x.

4.5.14 Linked task class

4.5.14.1 Linked task class overview

A linked task class is an application client class that represents a task that consists of a series of linked commands (see clause 5.7.2x).

Each instance of a linked task class represents the work associated with a series of linked commands. The first in a series of linked commands causes the creation of a linked task. The linked 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 linked command processing see 5.7.2.

4.5.14.2 CDB attribute

For a description of the CDB attribute contains a CDB as defined in 5.2x and SPC-3.

4.5.15 SCSI target device class

A SCSI target device class (see figure 17) is a SCSI device class that contains the:

a) level 1 hierarchical logical unit class (see 4.5.16) that contains the:
   A) logical unit class (see 4.5.20)
   B) well known logical unit class (see 4.5.25); or
   C) both.

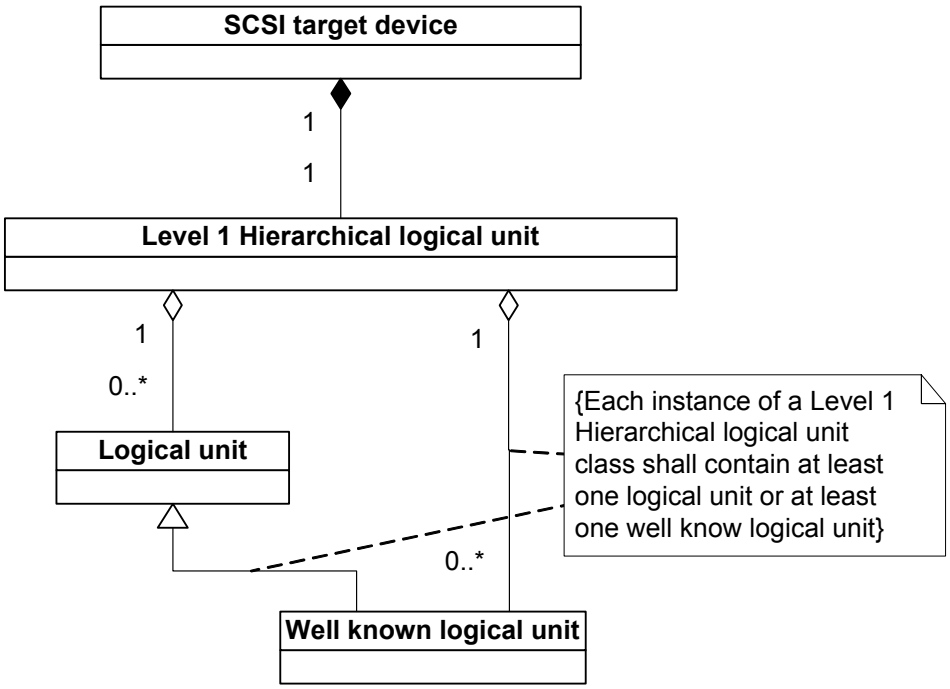


Figure 17 — SCSI target device class diagram

Each instance of a SCSI target device class shall contain the following objects:

a) one level 1 hierarchical logical unit that contains;
   A) at least one logical unit or well known logical unit;
   B) zero or more logical units; and
   C) zero or more well known logical units.

4.5.16 Level 1 hierarchical logical unit class

A level 1 hierarchical logical unit class (see figure 18) contains the:

a) logical unit class (see 4.5.20);
b) well known logical unit class (see 4.5.25); and
c) level 2 hierarchical logical unit class (see 4.5.17) that contains the:
   A) logical unit class;
   B) well known logical unit class; and
   C) level 3 hierarchical logical unit class (see 4.5.18) that contains the:
      a) logical unit class;
      b) well known logical unit class; and
      c) level 4 hierarchical logical unit class (see 4.5.19) that contains the:
         A) logical unit class; and
         B) well known logical unit class.

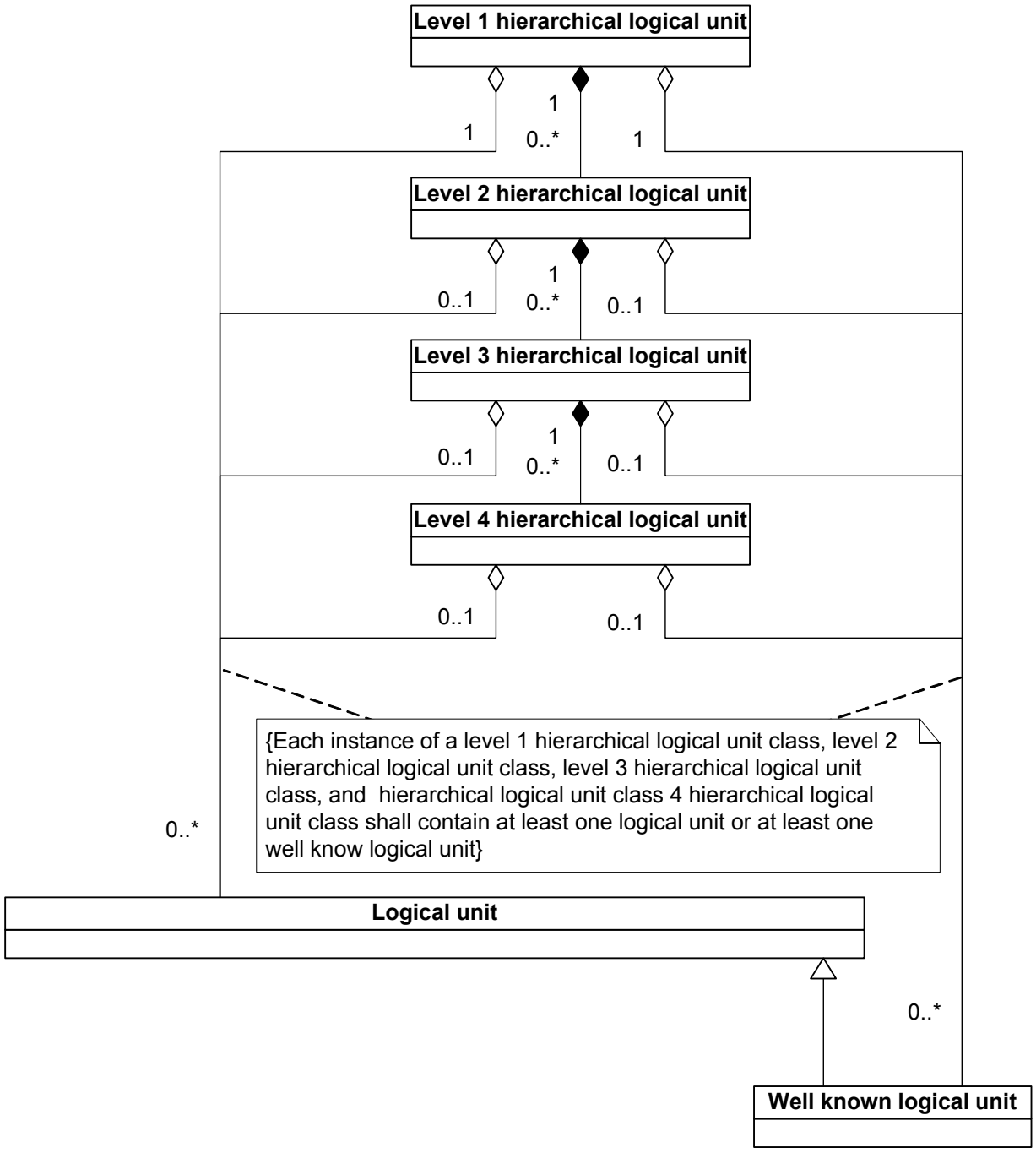


Figure 18 — Level 1 hierarchical logical unit class

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.

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 4.6.5);
b) Flat space addressing method (see 4.6.6);
c) Logical unit address method (see 4.6.7); or
d) Extended logical unit addressing method (see 4.6.8).

All peripheral device addresses, except LUN 0 (see 4.6.2), default to vendor specific values. All addressable entities, except well known logical units (see 4.5.25), 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 4.6.5 and 4.6.7.

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.

4.5.17 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.

All logical units and well known logical units contained within level 2 hierarchical logical unit shall have a dependent logical unit attribute (see 4.5.20.4).

4.5.18 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.

All logical units and well known logical units contained within level 3 hierarchical logical unit shall have a dependent logical unit attribute (see 4.5.20.4).

4.5.19 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.

All logical units and well known logical units contained within level 4 hierarchical logical unit shall have a dependent logical unit attribute (see 4.5.20.4).

4.5.20 Logical unit class

4.5.20.1 Logical unit class overview

A logical unit class (see figure 19) contains the:

a) device server class (see 4.5.21);
b) task manager class (see 4.5.22); and
c) task set class (see 4.5.23).

A logical unit class (see figure 19) may be substituted with the:

a) well known logical unit class (see 4.5.20.1); or
b) hierarchical logical unit class.

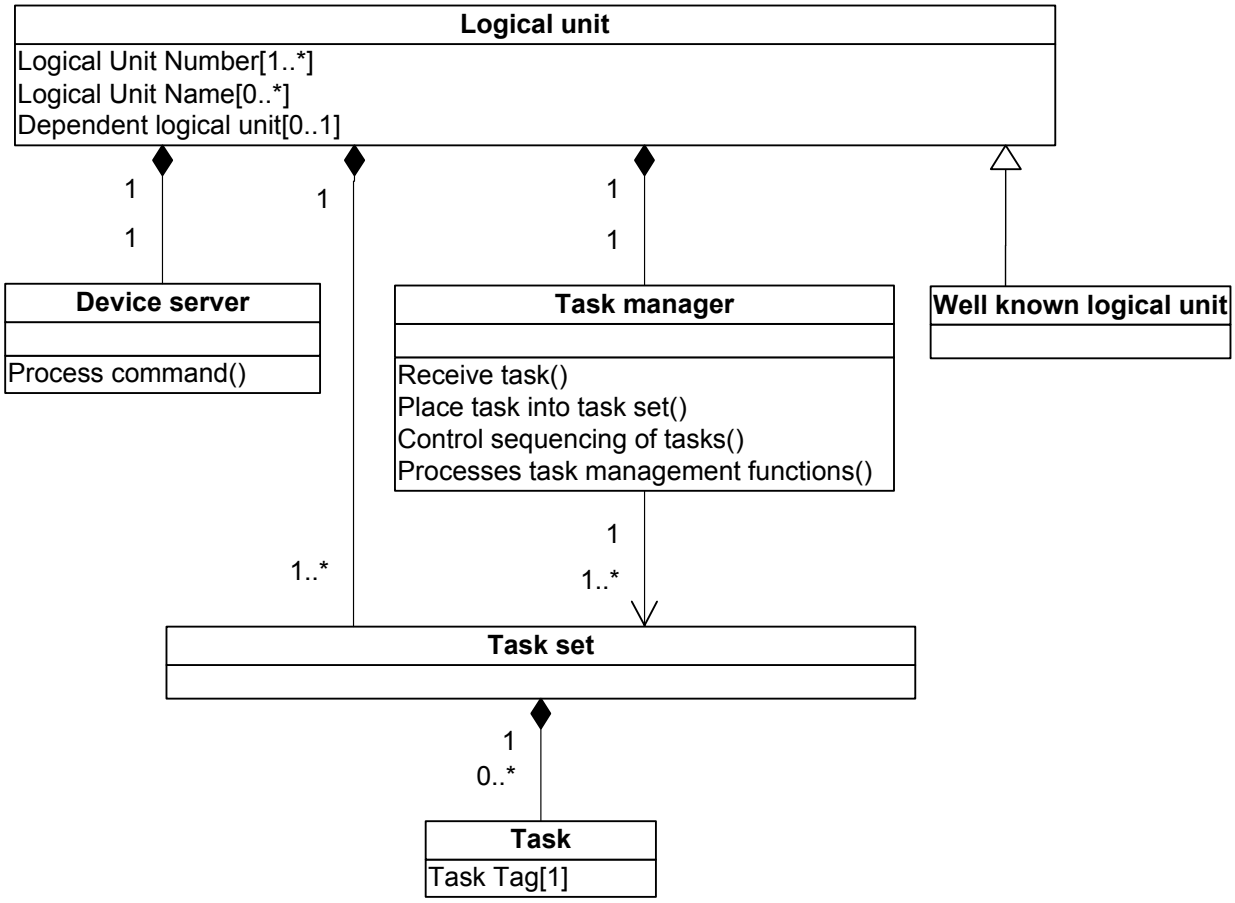


Figure 19 — Logical unit class diagram

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.

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.

4.5.20.2 Logical unit number attribute

A 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 3.1.22) in its composition, then all logical unit numbers within the scope of the SCSI target device shall have the format described in 4.6.4. 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 4.6.3.

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), 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 7.10.

4.5.20.3 Logical unit name attribute

A logical unit name attribute identifies a name (see 3.1.63) 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), 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.

4.5.20.4 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 4.6.4. 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 4.6.4; 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.

4.5.21 Device server class

The device server class processes commands.

4.5.22 Task manager class

The task manager class:

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

4.5.23 Task set class

A task set class contains a task class (see 4.5.24).

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 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).

4.5.24 Task class

4.5.24.1 Task class overview

A task class represents the work associated with a command or a group of linked commands. There shall be one task class for each task and linked task that the device server has not stated processing.

A task and a linked 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 or a group of linked commands;
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).

4.5.24.2 Task tag attribute

A task tag attribute identifies a command or a group of linked commands. The I_T_L_Q nexus representing a task or linked task includes a task tag, allowing many uniquely identified tagged tasks and linked 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 or linked task from a specific SCSI initiator port having identical values for the target port identifier, logical unit number, and task tag.

4.5.25 Well known logical unit class

A well known logical unit class is a logical unit class (see 4.5.20.1) with the additional characteristics defined in this subclause.

Well known logical units are addressed using the well known logical unit addressing method (see 4.6.9) of extended logical unit addressing (see 4.6.8). 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 7.10.

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 4.5.15).

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).

4.6 Logical unit numbers

4.6.1 Logical unit numbers overview

All logical unit number formats described in this standard are hierarchical in structure even when only a single level in that hierarchy is used. The HISUP bit shall be set to one in the standard INQUIRY data (see SPC-3) when any logical unit number format described in this standard is used. Non-hierarchical formats are outside the scope of this standard.

A logical unit number shall contain 64 bits or 16 bits, with the size being defined by the SCSI transport protocol. For SCSI transport protocols that define 16-bit logical unit numbers, the two bytes shall be formatted as described for the FIRST LEVEL ADDRESSING field (see table 5 in 4.6.4).

4.6.2 Minimum LUN addressing requirements

All SCSI devices shall support LUN 0 (i.e., 00000000 00000000h) or the REPORT LUNS well-known logical unit. For SCSI devices that support the hierarchical addressing model the LUN 0 or the REPORT LUNS well-known logical unit shall be the logical unit that an application client addresses to determine information about the SCSI target device and the logical units contained within the SCSI target device.

The responses to commands sent to unsupported logical units are defined in 5.8.4. The response to task management functions sent to unsupported logical units is defined in 7.1.

4.6.3 Single level logical unit number structure

Table 2 describes a single level subset of the format described in 4.6.4 for logical unit numbers 255 and below.

Table 2 — Single level logical unit number structure for logical unit numbers 255 and below

Bit / Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (00b)		BUS IDENTIFIER (00h)					
1	SINGLE LEVEL LUN (00h to FFh, inclusive)							
2	(MSB)	Null second level LUN (0000h)						
3								(LSB)
4	(MSB)	Null third level LUN (0000h)						
5								(LSB)
6	(MSB)	Null fourth level LUN (0000h)						
7								(LSB)

All logical unit number structure fields shall be zero except the SINGLE LEVEL LUN field (see table 2). The value in the SINGLE LEVEL LUN field shall be between 0 and 255, inclusive. The 00b in the ADDRESS METHOD field specifies peripheral device addressing (see 4.6.4) and the 00h in the BUS IDENTIFIER field specifies the current level (see 4.6.5).

Table 3 describes a single level subset of the format described in 4.6.4 for logical unit numbers 16 383 and below.

Table 3 — Single level logical unit number structure for logical unit numbers 16 383 and below

Bit / Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (01b)		(MSB)					
1	SINGLE LEVEL LUN (0000h to 3FFFh, inclusive)							(LSB)
2	(MSB)	Null second level LUN (0000h)						
3								(LSB)
4	(MSB)	Null third level LUN (0000h)						
5								(LSB)
6	(MSB)	Null fourth level LUN (0000h)						
7								(LSB)

All logical unit number structure fields shall be zero except the SINGLE LEVEL LUN field (see table 3). The value in the SINGLE LEVEL LUN field shall be between 0 and 16 383, inclusive. The 01b in the ADDRESS METHOD field specifies flat space addressing (see 4.6.6) at the current level.

If a SCSI target device contains 256 or fewer logical units, none of which are dependent logical units (see 4.5.20.4) or extended addressing logical units (see 4.6.8), then its logical units should be numbered 255 and below, and should have the format shown in table 2 (i.e., peripheral device addressing) but may have the format shown in table 3 (i.e., flat space addressing).

If a SCSI target device contains 16 384 or fewer logical units, none of which are dependent logical units or extended addressing logical units, then its logical units should be numbered 16 383 and below, and should have the format shown in table 3 (i.e., flat space addressing) but may have the format shown in table 2 (i.e., peripheral device addressing) for logical unit numbers that are less than 256.

4.6.4 Eight byte logical unit number structure

The eight byte logical unit number structure (see table 5) contains four levels of addressing fields. Each level shall use byte 0 and byte 1 to define the address and location of the SCSI device to be addressed on that level.

If the logical unit number specifies that the command is to be relayed to the next level then the current level shall use byte 0 and byte 1 of the eight byte logical unit number structure to determine the address of the SCSI device to which the command is to be sent. When the command is sent to the SCSI target device the eight byte logical unit number structure that was received shall be adjusted to create a new eight byte logical unit number structure (see table 4 and figure 20).

SCSI devices shall keep track of the addressing information necessary to transmit information back through all intervening levels to the task's originating SCSI initiator port

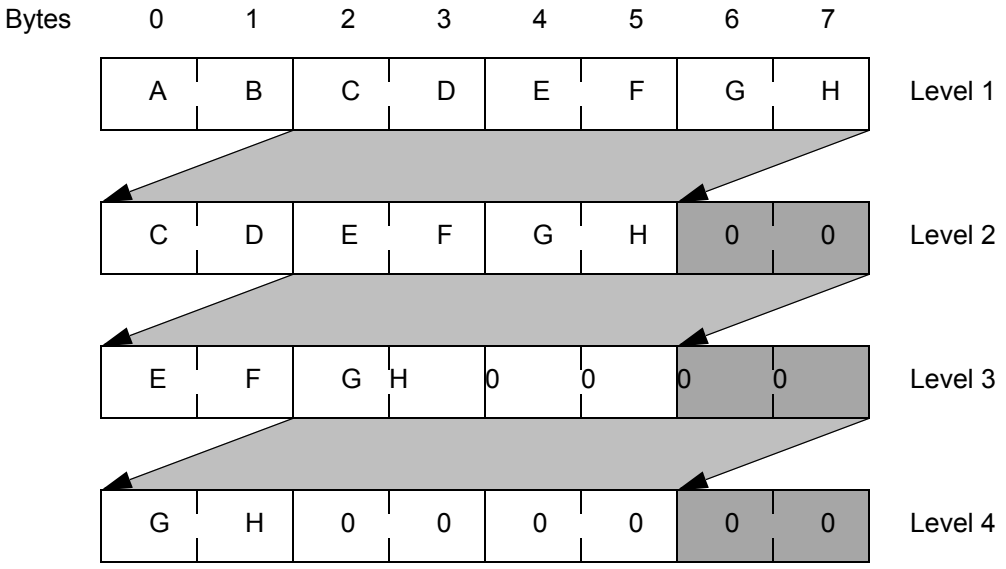


Figure 20 — Eight byte logical unit number structure adjustments

Table 4 — Eight byte logical unit number structure adjustments

Byte position		
Old		**New**
0 & 1	Moves to	Not Used
2 & 3	Moves to	0 & 1
4 & 5	Moves to	2 & 3
6 & 7	Moves to	4 & 5
N/A	zero fill	6 & 7

The eight byte logical unit number structure requirements as viewed from the application client are shown in table 5.

Table 5 — Eight byte logical unit number structure

Bit Byte	7	6	5	4	3	2	1	0
0	first level addressing							
1								
2	second level addressing							
3								
4	third level addressing							
5								
6	fourth level addressing							
7								

The FIRST LEVEL ADDRESSING field specifies the first level address of a SCSI device. See table 6 for a definition of the FIRST LEVEL ADDRESSING field.

The SECOND LEVEL ADDRESSING field specifies the second level address of a SCSI device. See table 6 for a definition of the SECOND LEVEL ADDRESSING field.

The THIRD LEVEL ADDRESSING field specifies the third level address of a SCSI device. See table 6 for a definition of the THIRD LEVEL ADDRESSING field.

The FOURTH LEVEL ADDRESSING field specifies the fourth level address of a SCSI device. See table 6 for a definition of the FOURTH LEVEL ADDRESSING field.

Table 6 — Format of addressing fields

Bit Byte	7	6	5	4	3	2	1	0
n-1	address method							
n	address method specific							

The ADDRESS METHOD field defines the contents of the ADDRESS METHOD SPECIFIC field. See table 7 for the address methods defined for the ADDRESS METHOD field. The ADDRESS METHOD field only defines address methods for entities that are directly addressable by an application client.

Table 7 — ADDRESS METHOD field values

Code	Description	Reference
00b	Peripheral device addressing method	4.6.5
01b	Flat space addressing method	4.6.6
10b	Logical unit addressing method	4.6.7
11b	Extended logical unit addressing method	4.6.8

4.6.5 Peripheral device addressing method

If the peripheral device addressing method (see table 8) is selected, the SCSI device should relay the received command or task management function to the addressed dependent logical unit. Any command that is not relayed to a dependent logical unit shall be terminated with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID COMMAND OPERATION CODE. If a task management function cannot be relayed to a dependent logical unit, a service response of SERVICE DELIVERY OR TARGET FAILURE shall be returned.

NOTE 2 - A SCSI device may filter (i.e., not relay) commands or task management functions to prevent operations with deleterious effects from reaching a dependent logical unit (e.g., a WRITE command directed to a logical unit that is participating in a RAID volume).

Table 8 — Peripheral device addressing

Bit Byte	7	6	5	4	3	2	1	0
n-1	ADDRESS METHOD (00b)		bus identifier					
n	target or lun							

The BUS IDENTIFIER field identifies the bus or path that the SCSI device shall use to relay the received command or task management function. The BUS IDENTIFIER field may use the same value encoding as the BUS NUMBER field (see 4.6.7) with the most significant bits set to zero. However, bus identifier zero shall specify that the

command or task management function is to be relayed to a logical unit within the SCSI device at the current level.

The TARGET OR LUN field specifies the address of the peripheral device (e.g., a SCSI device at the next level) to which the SCSI device shall relay the received command or task management function. The meaning and usage of the TARGET OR LUN field depends on whether the BUS IDENTIFIER field contains zero.

A BUS IDENTIFIER field of zero specifies a logical unit at the current level. This representation of a logical unit may be used either when the SCSI device at the current level does not use hierarchical addressing for assigning LUNs to entities or when the SCSI device at the current level includes entities that are assigned LUNs but are not attached to SCSI buses. When the BUS IDENTIFIER field contains zero, the command or task management function shall be relayed to the current level logical unit specified by the TARGET OR LUN field within or joined to the current level SCSI device.

A BUS IDENTIFIER field greater than zero represents a SCSI domain that connects a group of SCSI devices to the current level SCSI device. Each SCSI domain shall be assigned a unique bus identifier number from 1 to 63. These bus identifiers shall be used in the BUS IDENTIFIER field when assigning addresses to peripheral devices attached to the SCSI domains. When the BUS IDENTIFIER field is greater than zero, the command or task management function shall be relayed to the logical unit with the logical unit number zero within the SCSI target device specified in the TARGET OR LUN field located in the SCSI domain specified by the BUS IDENTIFIER field. The SCSI target device information in the TARGET OR LUN field is a mapped representation of a target port identifier.

The SCSI device located within the current level may be addressed by a BUS IDENTIFIER field and a TARGET OR LUN field of all zeros, also known as LUN 0 (see 4.6.2).

4.6.6 Flat space addressing method

The flat space addressing method (see table 9) specifies a logical unit at the current level.

The contents of all hierarchical structure addressing fields following a flat space addressing method addressing field shall be ignored.

Table 9 — Flat space addressing

Bit / Byte	7	6	5	4	3	2	1	0
n-1	ADDRESS METHOD (01b)		(MSB)					
n	lun							(LSB)

The LUN field specifies the current level logical unit.

4.6.7 Logical unit addressing method

If the logical unit addressing method (see table 10) is selected, the SCSI device should relay the received command or task management function to the addressed dependent logical unit. Any command that is not relayed to a dependent logical unit shall be terminated with a CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to INVALID COMMAND OPERATION CODE. If a task management function cannot be relayed to a dependent logical unit, a service response of SERVICE DELIVERY OR TARGET FAILURE shall be returned.

NOTE 3 - A SCSI device may filter (i.e., not relay) commands or task management functions to prevent operations with deleterious effects from reaching a dependent logical unit (e.g., a WRITE command directed to a logical unit that is participating in a RAID volume).

The contents of all hierarchical structure addressing fields following a logical unit addressing method addressing field shall be ignored.

Table 10 — Logical unit addressing

Bit Byte	7	6	5	4	3	2	1	0
n-1	ADDRESS METHOD (10b)		target					
n	bus number			lun				

The TARGET field, BUS NUMBER field, and LUN field address the logical unit to which the received command or task management function shall be relayed. The command or task management function shall be relayed to the logical unit specified by the LUN field within the SCSI target device specified by the TARGET field located on the bus specified by the BUS NUMBER field. The value in the LUN field shall be placed in the least significant bits of the SINGLE LEVEL LUN field in a single level logical unit number structure for logical unit numbers 255 and below (see 4.6.1). The TARGET field contains a mapped representation of a target port identifier.

4.6.8 Extended logical unit addressing

Extended logical unit addressing (see table 11) specifies a logical unit at the current level.

Extended logical unit addressing builds on the formats defined for dependent logical units (see 4.5.16) but may be used by SCSI devices having single level logical unit structure. In dependent logical unit addressing, the logical unit information at each level fits in exactly two bytes. Extended logical unit addresses have sizes of two bytes, four bytes, six bytes, or eight bytes.

The contents of all hierarchical structure addressing fields following an extended logical unit addressing method addressing field shall be ignored.

Extended logical units are identified by the ADDRESS METHOD field (see table 7 in 4.6.4) in the same manner as is the case for dependent logical units. An ADDRESS METHOD field value of 11b specifies the extended logical unit addressing method.

Table 11 — Extended logical unit addressing

Bit Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		length		extended address method			
m	extended address method specific							

The LENGTH field (see table 12) specifies the length of the EXTENDED ADDRESS METHOD SPECIFIC field.

Table 12 — LENGTH field values and related sizes

Value	Size in bytes of		Reference
	EXTENDED ADDRESS METHOD SPECIFIC field	Extended logical unit addressing format	
00b	1	2	table 13
01b	3	4	table 14
10b	5	6	table 15
11b	7	8	table 16

Table 13, table 14, table 15, and table 16 show the four extended logical unit addressing formats.

Table 13 — Two byte extended logical unit addressing format

Bit / Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (00b)		extended address method			
n+1	extended address method specific							

Table 14 — Four byte extended logical unit addressing format

Bit / Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (01b)		extended address method			
n+1	(MSB)							
	extended address method specific							
n+3								(LSB)

Table 15 — Six byte extended logical unit addressing format

Bit / Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (10b)		extended address method			
n+1	(MSB)							
	extended address method specific							
n+5								(LSB)

Table 16 — Eight byte extended logical unit addressing format

Bit Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (11b)		LENGTH (11b)		extended address method			
1	(MSB)							
				extended address method specific				
7								(LSB)

The EXTENDED ADDRESS METHOD field combined with the LENGTH field (see table 17) specifies the type and size of extended logical unit address found in the EXTENDED ADDRESS METHOD SPECIFIC field.

Table 17 — Logical unit extended addressing

EXTENDED ADDRESS METHOD Code(s)	LENGTH Code(s)	Description	Reference
0h	00b - 11b	Reserved	
1h	00b	Well known logical unit	4.6.9
1h	01b - 11b	Reserved	
2h - Eh	00b - 11b	Reserved	
Fh	00b - 10b	Reserved	
Fh	11b	Logical unit not specified	4.6.10

4.6.9 Well known logical unit addressing

A SCSI target device may support zero or more well known logical units (see 4.5.25). A single SCSI target device shall only support one instance of each supported well known logical unit. All well known logical units within a SCSI target device shall be accessible from all SCSI target ports contained within the SCSI target device.

Well known logical units are addressed using the well known logical unit extended address format (see table 18).

Table 18 — Well known logical unit extended address format

Bit Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (00b)		EXTENDED ADDRESS METHOD (1h)			
n+1	W-LUN							

The W-LUN field specifies the well known logical unit to be addressed (see SPC-3).

4.6.10 Logical unit not specified addressing

Logical unit not specified addressing (see table 19) shall be used to indicate that no logical unit of any kind is specified.

Table 19 — Logical unit not specified extended address format

Bit Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (11b)		LENGTH (11b)		EXTENDED ADDRESS METHOD (Fh)			
1	FFh							
2	FFh							
3	FFh							
4	FFh							
5	FFh							
6	FFh							
7	FFh							

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 20 maps the nexus object to other identifier objects.

Table 20 — Mapping nexus to SAM-2 identifiers

Nexus	Identifiers contained in nexus	Reference
I_T	Initiator Port Identifier Target Port Identifier	4.5.9 4.5.15
I_T_L	Initiator Port Identifier Target Port Identifier Logical Unit Number	4.5.9 4.5.15 4.5.20
I_T_L_Q	Initiator Port Identifier Target Port Identifier Logical Unit Number Task Tag	4.5.9 4.5.15 4.5.20 4.5.24.2

4.8 SCSI ports

4.8.1 SCSI port configurations

A SCSI device may contain only SCSI target ports, only SCSI initiator ports, or any combination of ports. Some of the port configurations possible for a SCSI device are shown in figure 21.

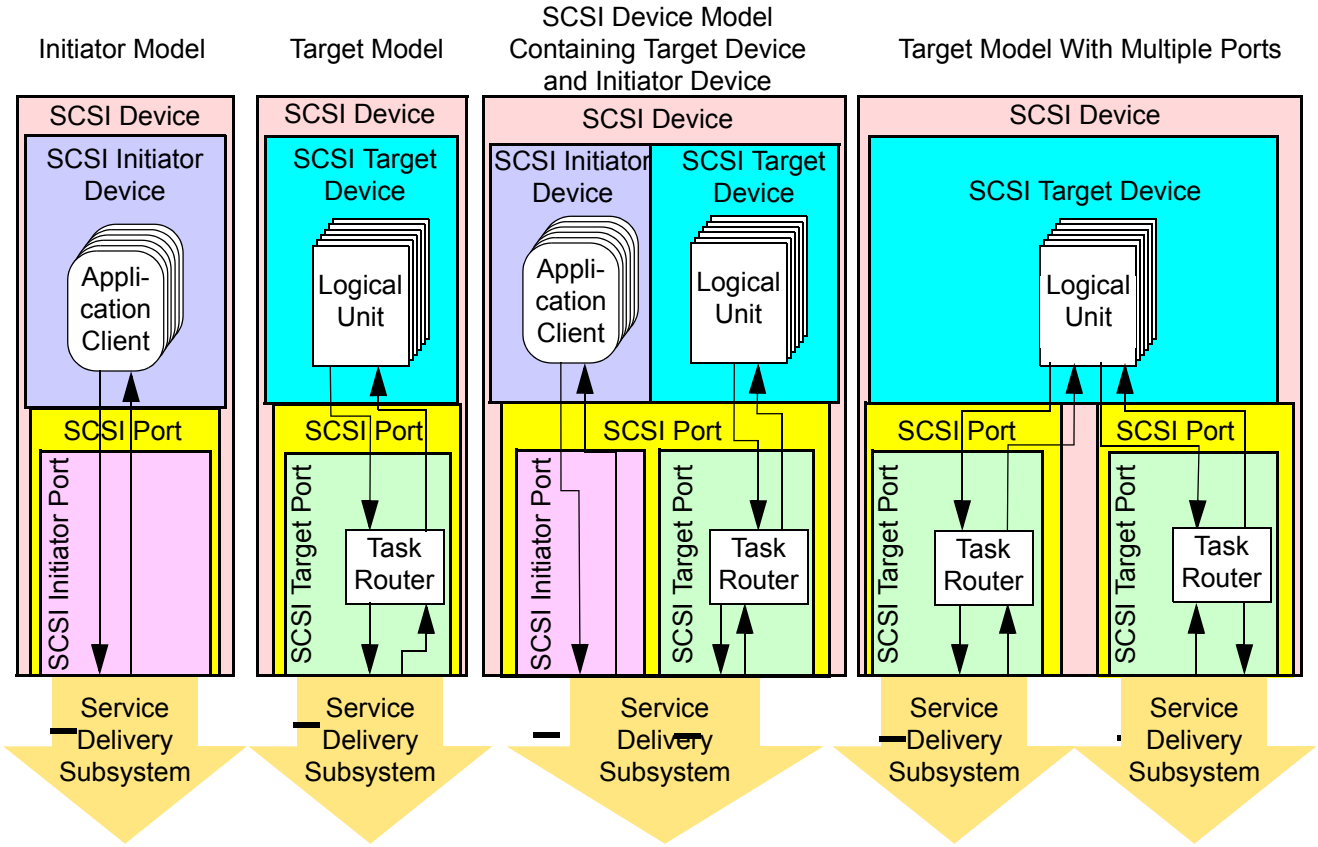


Figure 21 — SCSI device functional models

4.8.2 SCSI devices with multiple ports

The model for a SCSI device with multiple ports is a single:

a) SCSI target device (see 4.5.15) with multiple SCSI target ports;
b) SCSI initiator device (see 4.5.9) with multiple SCSI initiator ports; or
c) SCSI device containing a SCSI initiator device and a SCSI target device, and multiple SCSI ports.

The SCSI identifiers representing the SCSI ports shall meet the requirements for initiator port identifiers (see 4.5.9) or target port identifiers (see 4.5.15). How a multiple port SCSI device is viewed by counterpart SCSI devices in the SCSI domain also depends on whether a SCSI initiator port is examining a SCSI target port, or a SCSI target port is servicing a SCSI initiator port.

4.8.3 Multiple port SCSI target device structure

Figure 22 shows the structure of a SCSI target device with multiple SCSI ports each containing a SCSI target port. Each SCSI target port contains a task router that is shared by a collection of logical units. Each logical unit contains a single task manager and a single device server.

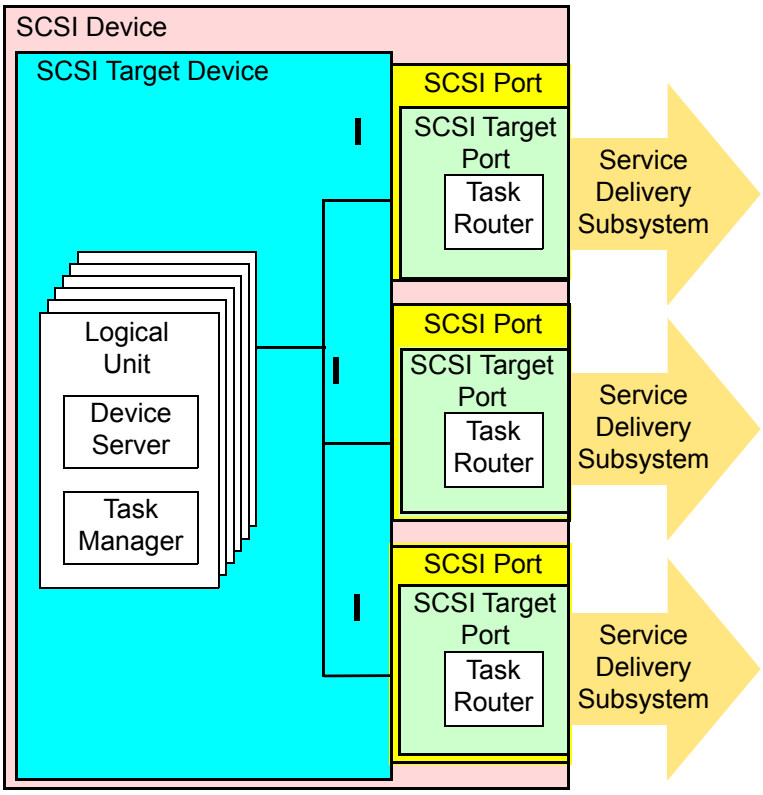


Figure 22 — Multiple port target SCSI device structure model

Two-way communications shall be possible between all logical units and all SCSI target ports, however, communications between any logical unit and any SCSI target port may be inactive. Two-way communications shall be available between each task manager and all task routers. Each SCSI target port shall accept commands sent to LUN 0 or the REPORT LUNS well-known logical unit and the task router shall route them to a device server for processing. The REPORT LUNS commands (see SPC-3) shall be accepted by the logical unit with the logical unit number zero or the REPORT LUNS well-known logical unit from any SCSI target port and shall return the logical unit inventory available via that SCSI target port. The availability of the same logical unit through multiple SCSI target ports is discovered by matching logical unit name values in the INQUIRY command Device Identification VPD page (see SPC-3).

4.8.4 Multiple port SCSI initiator device structure

Figure 23 shows the structure of a SCSI initiator device with multiple SCSI ports each containing a SCSI initiator port. Each SCSI initiator port is shared by a collection of application clients.

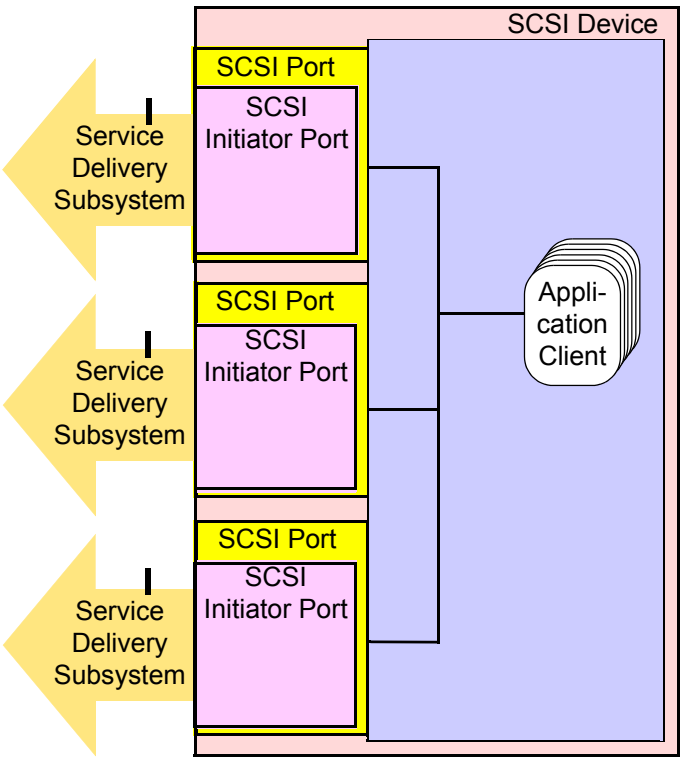


Figure 23 — Multiple port SCSI initiator device structure model

Two-way communications shall be possible between an application client and its associated SCSI initiator port. This standard does not specify or require the definition of any mechanisms by which a SCSI target device would have the ability to discover that it is communicating with multiple ports on a single SCSI initiator device. In those SCSI transport protocols where such mechanisms are defined, they shall not have any effect on how commands are processed (e.g., reservations shall be handled as if no such mechanisms exist).

4.8.5 Multiple port SCSI device structure

Figure 24 shows the structure of a SCSI device containing a SCSI target device and a SCSI initiator device, and multiple SCSI ports. Each SCSI port contains a SCSI target port and a SCSI initiator port. This SCSI device may also contain SCSI ports that only contain a SCSI target port or a SCSI initiator port. Each SCSI port consists of a SCSI target port containing a task router and a SCSI initiator port and is shared by a collection of logical units and application clients. Each logical unit contains a task manager and a device server.

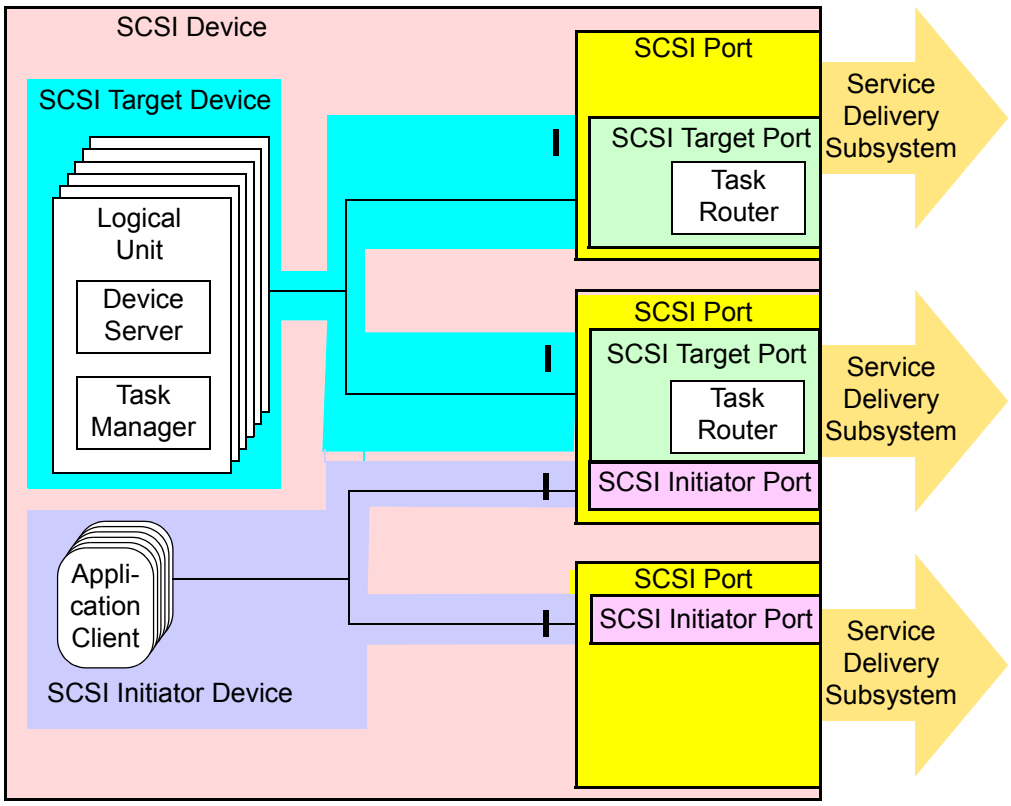


Figure 24 — Multiple port SCSI device structure model

Two-way communications shall be possible between all logical units and all SCSI target ports, however, communications between any logical unit and any SCSI target port may be inactive. Two-way communications shall be possible between an application client and its associated SCSI initiator port. Each SCSI target port shall accept commands sent to LUN 0 or the REPORT LUNS well-known logical unit and the task router shall route them to a device server for processing. The REPORT LUNS commands (see SPC-3) shall be accepted by the logical unit with the logical unit number zero or the REPORT LUNS well-known logical unit from any SCSI target port and shall return the logical unit inventory available via that SCSI target port. The availability of the same logical unit through multiple SCSI target ports is discovered by matching logical unit name values in the INQUIRY command Device Identification VPD page (see SPC-3).

This standard does not specify or require the definition of any mechanisms by which a SCSI target device would have the ability to discover that it is communicating with multiple SCSI ports that also contain a SCSI initiator port on a single SCSI device. In those SCSI transport protocols where such mechanisms are defined, they shall not have any effect on how commands are processed (e.g., reservations shall be handled as if no such mechanisms exist).

4.8.6 SCSI initiator device view of a multiple port SCSI target device

A SCSI target device may be connected to multiple SCSI domains such that a SCSI initiator port is only able to communicate with its logical units using a single SCSI target port. However, SCSI target devices with multiple SCSI ports may be configured where application clients have the ability to discover that one or more logical units are accessible via multiple SCSI target ports. Figure 25 and figure 26 show two examples of such configurations.

Figure 25 shows a SCSI target device with multiple SCSI ports each containing a SCSI target port participating in a single SCSI domain with two SCSI initiator devices. There are three SCSI devices, one of which has two SCSI

target ports, and two of which have one SCSI initiator port each. There are two target port identifiers and two initiator port identifiers in this SCSI domain. Using the INQUIRY command Device Identification VPD page (see SPC-3), the application clients in each of the SCSI initiator devices have the ability to discover if the logical units in the SCSI target devices are accessible via multiple SCSI target ports and map the configuration of the SCSI target device.

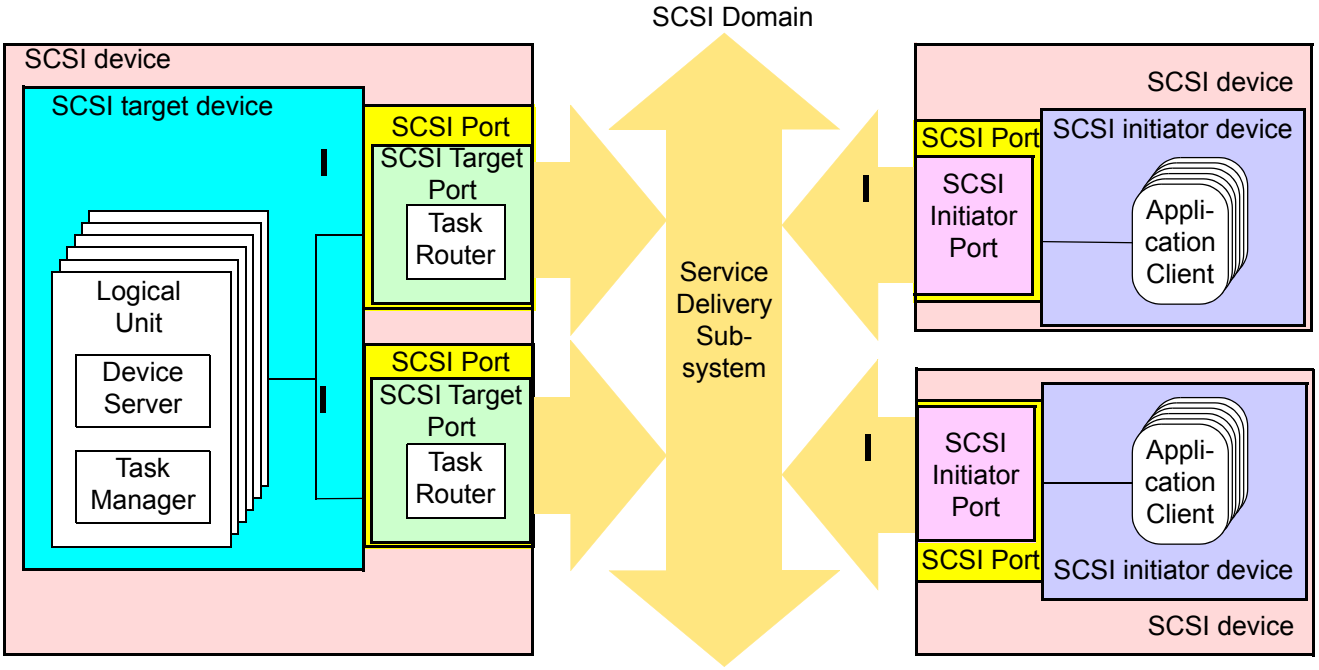


Figure 25 — SCSI target device configured in a single SCSI domain

Figure 26 shows a SCSI target device with multiple SCSI ports each containing a SCSI target port participating in two SCSI domains and a SCSI initiator device with multiple SCSI ports each containing a SCSI initiator port participating in the same two SCSI domains. There is one SCSI target device with two SCSI target ports and one SCSI initiator device with two SCSI initiator ports. There is one target port identifier and one initiator port identifier in each of the two SCSI domains. Using the INQUIRY command Device Identification VPD page (see SPC-3), the application clients in the SCSI initiator device have the ability to discover that logical units in the SCSI target device are accessible via multiple SCSI initiator ports and multiple SCSI target ports and map the configuration. However, application clients may not be able to distinguish between the configuration shown in figure 26 and the configuration shown in figure 27.

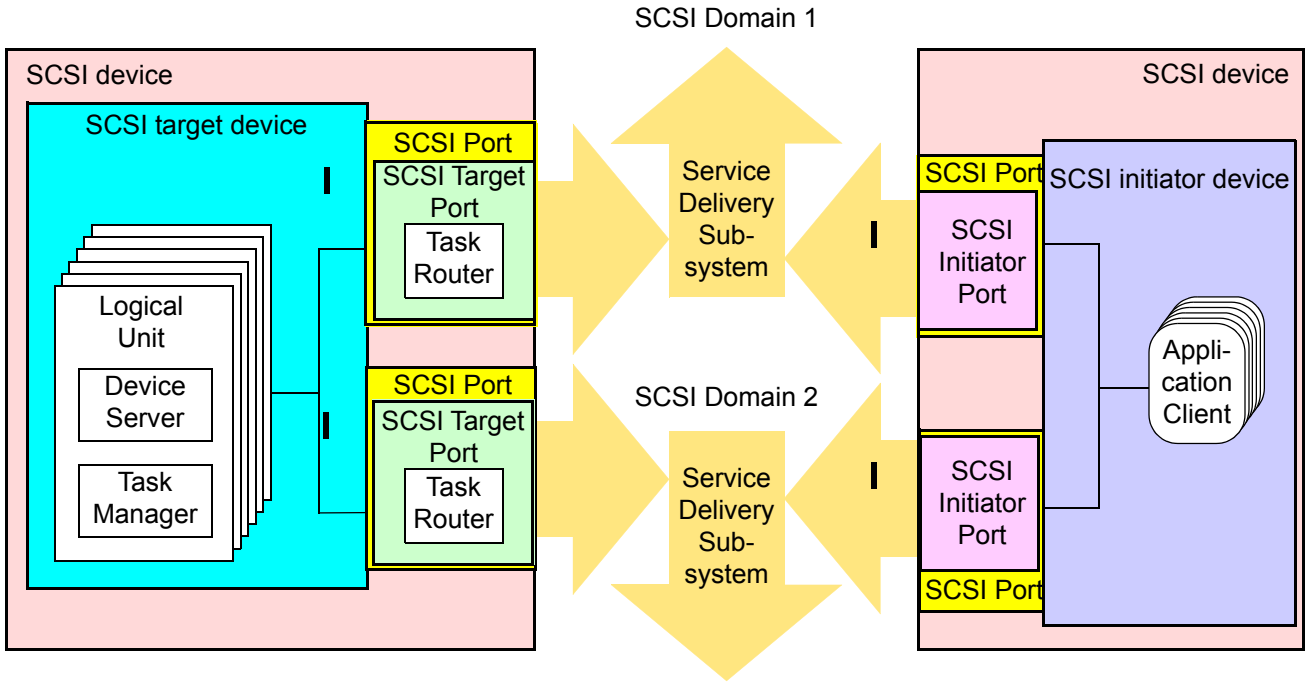


Figure 26 — SCSI target device configured in multiple SCSI domains

Figure 27 shows the same configuration as figure 26 except that the two SCSI domains have been replaced by a single SCSI domain.

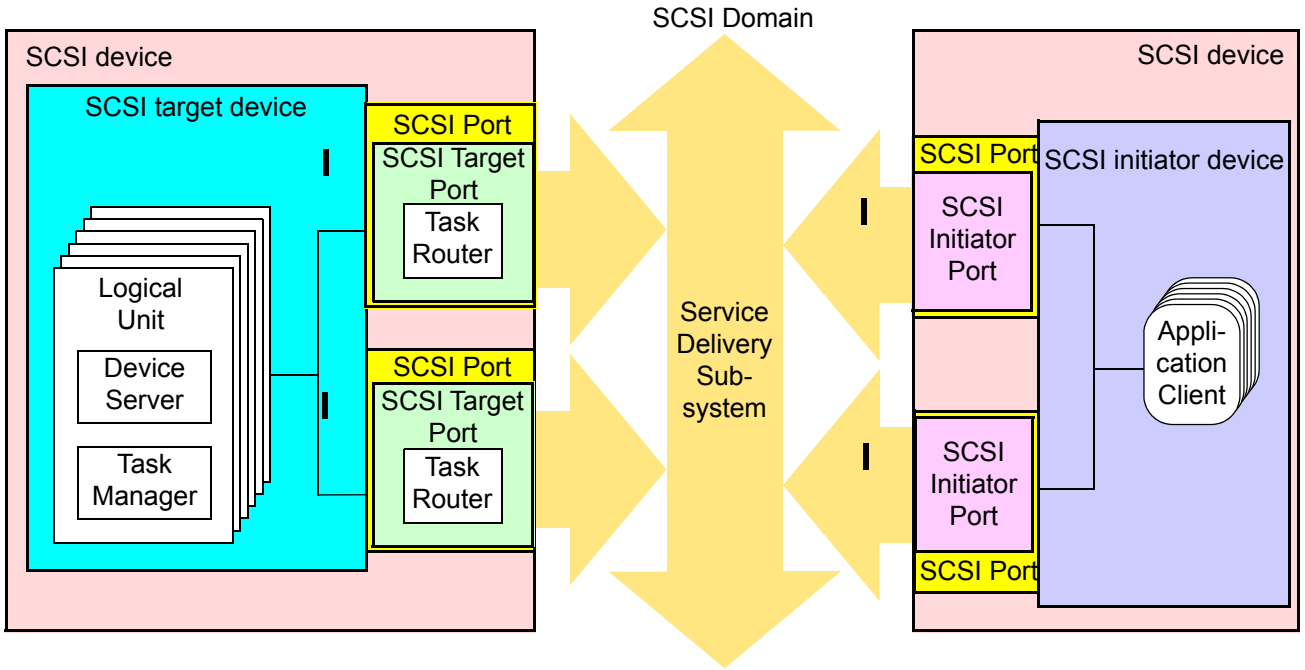


Figure 27 — SCSI target device and SCSI initiator device configured in a single SCSI domain

This model for application client determination of multiple SCSI target port configurations relies on information that is available only to the application clients via commands.

4.8.7 SCSI target device view of a multiple port SCSI initiator device

This standard does not require a SCSI target device to have the ability to detect the presence of a SCSI initiator device with multiple SCSI initiator ports. Therefore, a SCSI target device handles a SCSI initiator device with multiple SCSI initiator ports exactly as it would handle multiple separate SCSI initiator devices (e.g., a SCSI target device handles the configurations shown in figure 26 and figure 27 in exactly the same way it handles the configuration shown in figure 25).

NOTE 4 - The implications of this view of a SCSI initiator device are more far reaching than are immediately apparent (e.g., after a SCSI initiator device makes a persistent exclusive access reservation via one SCSI initiator port, access is denied to the other SCSI initiator port(s) on that same SCSI initiator device).

4.9 The SCSI model for distributed communications

The SCSI model for communications between distributed objects is based on the technique of layering as shown in figure 28.

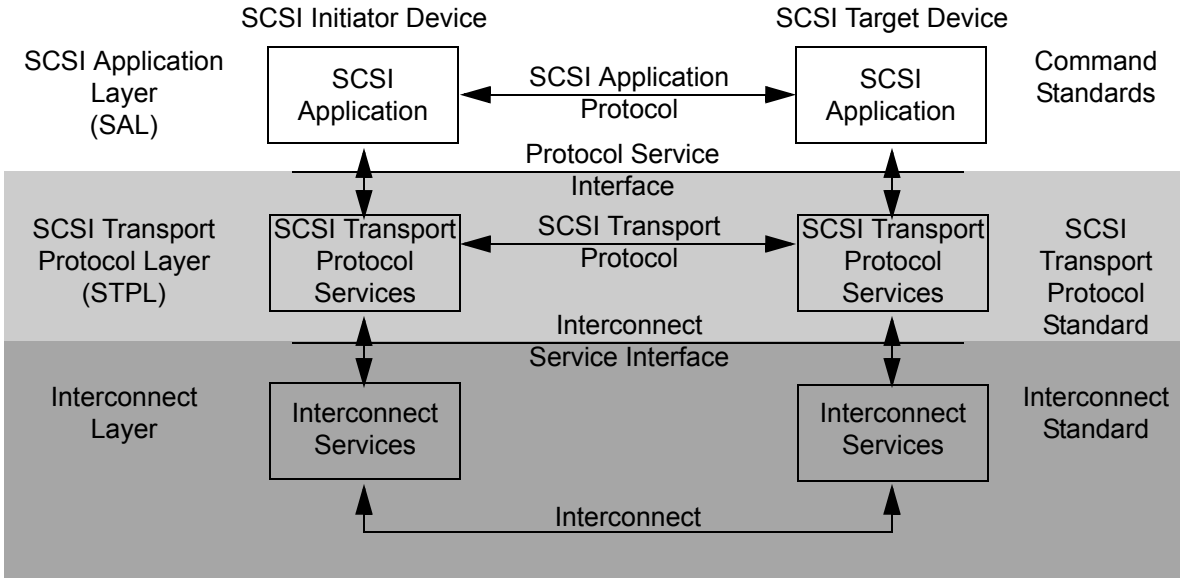


Figure 28 — Protocol service reference model

The layers comprising this model and the specifications defining the functionality of each layer are denoted by horizontal sequences. A layer consists of peer entities that communicate with one another by means of a protocol. Except for the interconnect layer, such communication is accomplished by invoking services provided by the adjacent layer. The following layers are defined:

SCSI application layer (SAL): Contains the clients and servers that originate and process SCSI I/O operations by means of a SCSI application protocol.
SCSI transport protocol layer (STPL): Consists of the services and protocols through which clients and servers communicate.
Interconnect layer: Comprised of the services, signaling mechanism and interconnect subsystem used for the physical transfer of data from sender to receiver. In the SCSI model, the interconnect layer is known as the service delivery subsystem.

The set of SCSI transport protocol services implemented by the service delivery subsystem identify external behavioral requirements that apply to SCSI transport protocol standards. While these SCSI transport protocol services may serve as a guide for designing reusable software or firmware that is adaptable to different SCSI

transport protocols, there is no requirement for an implementation to provide the service interfaces specified in this standard.

The SCSI transport protocol service interface is defined in this standard in representational terms using SCSI transport protocol services. The SCSI transport protocol service interface implementation is defined in each SCSI transport protocol standard. The interconnect service interface is described as appropriate in each SCSI transport protocol standard.

Interactions between the SAL and STPL are defined with respect to the SAL and may originate in either layer. An outgoing interaction is modeled as a procedure call invoking an STPL service. An incoming interaction is modeled as a procedure call invoked by the STPL.

All procedure calls may be accompanied by parameters or data. Both types of interaction are described using the notation for procedures specified in 3.6.2. In this standard, input arguments are defined relative to the layer receiving an interaction (i.e., an input is a argument supplied to the receiving layer by the layer initiating the interaction).

The following types of service interactions between layers are defined:

a) SCSI transport protocol service request procedure calls from the SAL invoking a service provided by the STPL;
b) SCSI transport protocol service indication procedure calls from the STPL informing the SAL that an asynchronous event has occurred (e.g., the receipt of a peer-to-peer protocol transaction);
c) SCSI transport protocol service response procedure calls to the STPL invoked by the SAL in response to a SCSI transport protocol service indication. A SCSI transport protocol service response may be invoked to return a reply from the invoking SAL to the peer SAL; and
d) SCSI transport protocol service confirmation procedure calls from the STPL notifying the SAL that a SCSI transport protocol service request has completed, has been terminated, or has failed to transit the interconnect layer. A confirmation may communicate parameters that indicate the completion status of the SCSI transport protocol service request or any other status. A SCSI transport protocol service confirmation may be used to convey a response from the peer SAL.

The services provided by an STPL are either confirmed or unconfirmed. A SAL service request invoking a confirmed service always results in a confirmation from the STPL.

Figure 29 shows the relationships between the four SCSI transport protocol service types.

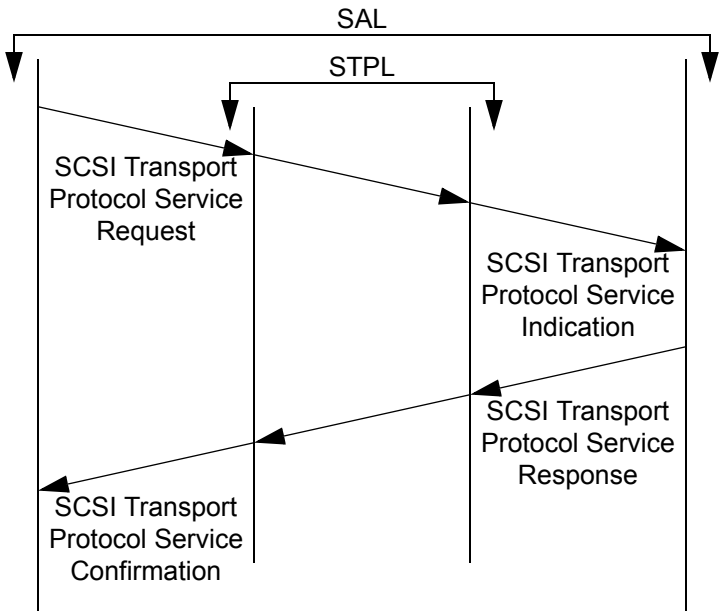


Figure 29 — SCSI transport protocol service mode

Figure 30 shows how SCSI transport protocol services may be used to process a client-server request-response transaction at the SCSI application layer.

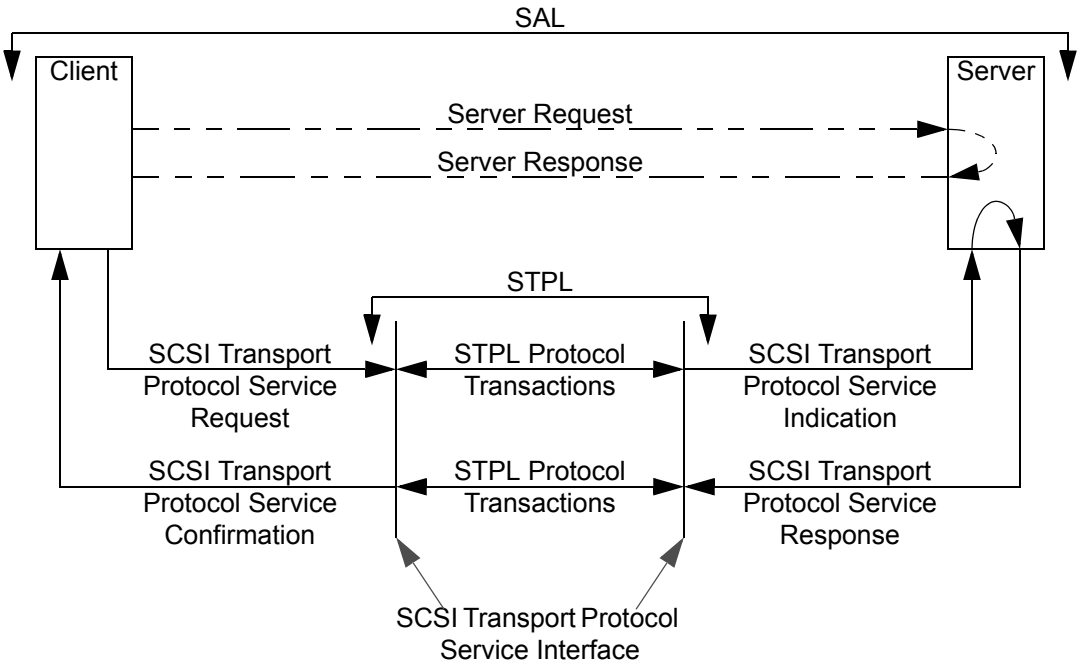


Figure 30 — Request-Response SAL transaction and related STPL services

The dashed lines in figure 30 show a SCSI application protocol transaction as it may appear to sending and receiving entities within the client and server. The solid lines in figure 30 show the corresponding SCSI transport protocol services and STPL transactions that are used to transport the data.

When a device server invokes a data transfer SCSI transport protocol service, the interactions required to transfer the data do not involve the application client. Only the STPL in the SCSI device that also contains the application client is involved. Figure 31 shows the relationships between the SCSI transport protocol service types involved in a data transfer request.

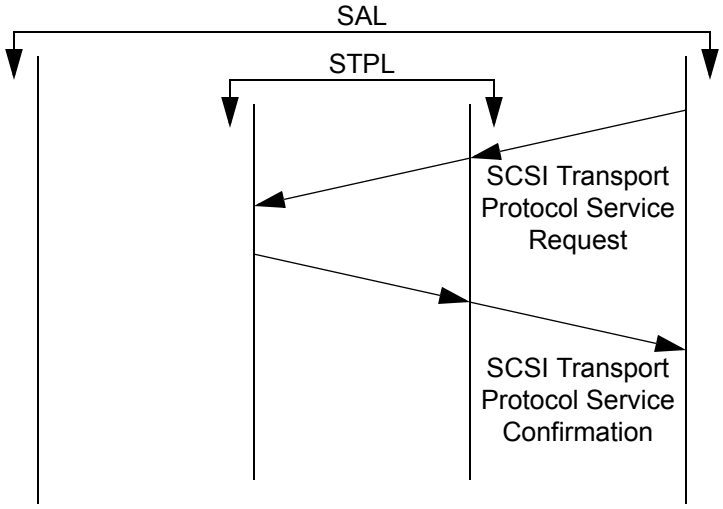
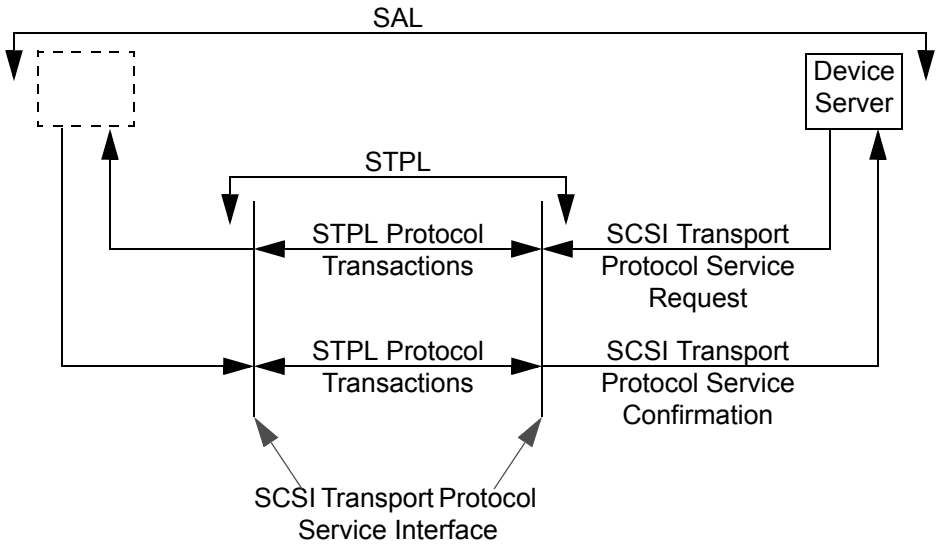


Figure 31 — SCSI transport protocol service model for data transfers

Figure 32 shows how SCSI transport protocol services may be used to process a device server data transfer transaction.



Note: The dotted box represents a memory access function provided by the SCSI initiator device whose definition is outside the scope of this standard.

Figure 32 — Device server data transfer transaction and related STPL services

When a device server invokes a Terminate Data Transfer SCSI transport protocol service, the interactions required to complete the service do not involve the SCSI Transport Protocol Service Interface or the application client. Only the STPL in the SCSI device that also contains the device server is involved. Figure 33 shows the relationships between the SCSI transport protocol service types involved in a Terminate Data Transfer request.

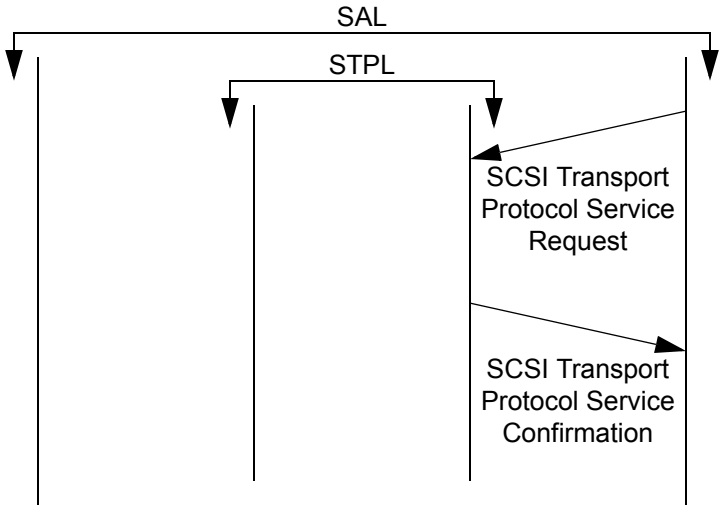


Figure 33 — SCSI transport protocol service model for Terminate Data Transfer

Figure 34 shows how SCSI transport protocol services may be used to process a device server Terminate Data Transfer transaction.

.

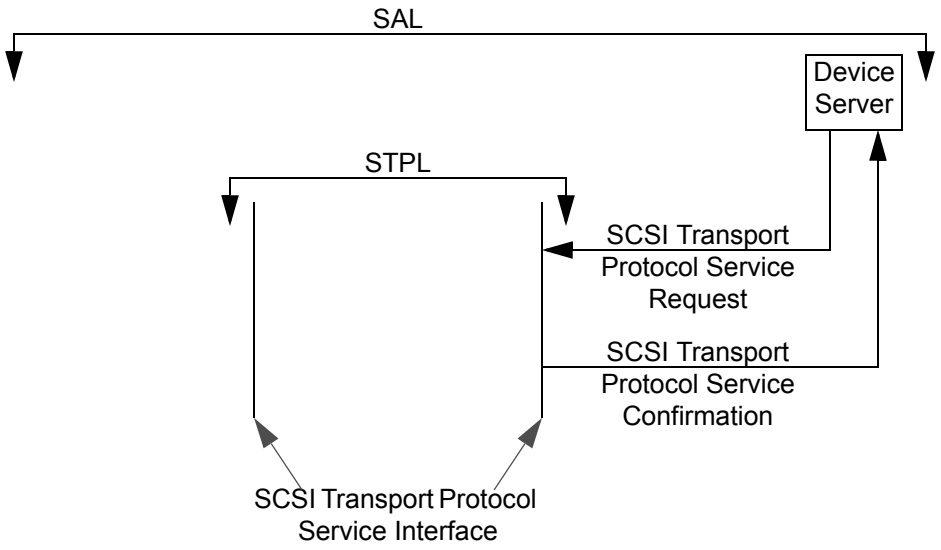


Figure 34 — Device server Terminate Data Transfer transaction and related STPL services