Working Draft American National Standard

Project T10/1683-D

Revision 11
18 May 2007

Information technology - SCSI Architecture Model - 4

T10 Technical Editor: George Penokie
IBM Corporation
MS: 49C
3605 Highway 52 N
Rochester, MN 55901
USA

Telephone: 507-253-5308
Email: gop@us.ibm.com

Reference number
ISO/IEC 14776-*:200x**
ANSI INCITS.*:200x**

Printed 10:45 AM Thursday 17 May 2007

Points of Contact

International Committee for Information Technology Standards (INCITS) T10 Technical Committee

T10 Chair
John B. Lohmeyer
LSI Logic
4420 Arrows West Drive
Colorado Springs, CO 80907-3444
USA

Telephone: (719) 533-7560
Email: lohmeyer@t10.org

T10 Vice-Chair
George O. Penokie
IBM Corporation
MS: 49C
3605 Highway 52 N
Rochester, MN 55901
USA

Telephone:(507) 253-5208
Email: gop@us.ibm.com

T10 Web Site: http://www.t10.org

T10 E-mail reflector:
Server: majordomo@t10.org
To subscribe send e-mail with ‘subscribe’ in message body
To unsubscribe send e-mail with ‘unsubscribe’ in message body

INCITS Secretariat
Suite 200
1250 Eye Street, NW
Washington, DC 20005
USA

Telephone:	202-737-8888
Web site:	http://www.incits.org
Email:	incits@itic.org

Information Technology Industry Council
Web site: http://www.itic.org

Document Distribution
INCITS Online Store
managed by Techstreet
1327 Jones Drive
Ann Arbor, MI 48105
USA

Web site:	http://www.techstreet.com/incits.html
Telephone:	(734) 302-7801 or (800) 699-9277

Global Engineering Documents, an IHS Company
15 Inverness Way East
Englewood, CO 80112-5704
USA

Web site:	http://global.ihs.com
Telephone:	(303) 397-7956 or (303) 792-2181 or (800) 854-7179

American National Standard
for Information Technology

SCSI Architecture Model - 4

Secretariat
Information Technology Industry Council

Approved mm.dd.yy

American National Standards Institute, Inc.

ABSTRACT

This standard specifies the SCSI Architecture Model. The purpose of the architecture is to provide a common basis for the coordination of SCSI standards and to specify those aspects of SCSI I/O system behavior that are independent of a particular technology and common to all implementations.

Published by
American National Standards Institute
11 W. 42nd Street, New York, New York 10036

Revision Information

R.1 Revision SAM-4r00 (14 January 2005)

First release of SAM-4. Same content as SAM-3 revision 14.

a) Contains ANSI edit changes to SAM-3.

R.2 Revision SAM-4r01 (28 January 2005)

The following T10 approved proposals were incorporated in SAM-4 in this revision:

a) 04-297r2 - SAM-3 SAS-1.1 FCP-3 - Retry Delay; and
b) 04-372r2 - I_T NEXUS LOSS task management function.

No other changes were made in revision 1.

R.3 Revision SAM-4r02 (16 March 2005)

The following T10 approved proposals were incorporated in SAM-4 in this revision:

a) 04-376r1 - SAM-4 SAS-1.1 Task priority cleanup;
b) 05-081r1 - SAM-4 task management function lifetime; and
c) 05-107r1 - SAS-1.1 SAM-4 Overlapped tag handling.

The following editorial changes were incorporated as a result of an E-mail dated 2/22/2005:

4.8 In "The task manager controls the sequencing of one or more tasks within a logical unit. The task manager also processes the task management functions specified in clause 7. There is one task manager per logical unit." change the middle sentence to "The task manager also processes task management functions (see clause 7)." There could be additional ones not specified in clause 7 (e.g. iSCSI and parallel SCSI have/had some).

4.13.3 well-know s/b well-known

4.13.3 "a single task manager and a device server" s/b "...and a <single> device server".

4.13.5 missing period at end of first paragraph

4.15 missing period in figure 31 sentence

5.8.1 figure 33 vs. 7.10 figure 37

figure 33 uses these labels:

initiator
application client task
task <<<---- compare
activity
target

figure 37 uses these labels:

initiator
application client task
activity
task manager <<<---- compare
activity
target

In figure 37 task manager s/b "task management function" to match.

All the bare initiator/targets should be fixed and consistent. In figure 34, "Device Server" is used where figure 33/37 use "Target". Change target to “Device Server” in figures 33 and 34. Change “Initiator” to “Application Client” in figures 33, 34, and 37. Change “Target” to “Task Manager” in figure 37.

7.1 "FUNCTION SUCCEEDED: An optional task manager response indicating that the requested function is supported and completed successfully. This task manager response shall only be used by functions that require notification of success."

I think people are still confused about the difference between FUNCTION COMPLETE and FUNCTION SUCCEEDED. After success add "(e.g., QUERY TASK)" so it stands out a bit more as the oddball.

R.4 Revision SAM-4r03 (20 September 2005)

The following T10 approved proposal was incorporated in SAM-4 in this revision:

a) 05-301r0 - SAM-4 iSCSI allows portal group tag of zero.

R.5 Revision SAM-4r04 (15 November 2005)

The following T10 approved proposal was incorporated in SAM-4 in this revision:

a) 05-370r2 - SAS-2, SAM-4, SPC-4, NOTIFY (POWER FAILURE EXPECTED) Fixes; and
b) 05-376r5 - SAM-4: Addressing more than 16384 logical units.

R.6 Revision SAM-4r05 (19 January 2006)

The following T10 approved proposal was incorporated in SAM-4 in this revision:

a) 06-003r1 - SAM-4 LUN representation format; and
b) 06-025r1 - SAM-4 More initiator port to I_T nexus changes.

R.7 Revision SAM-4r06 (23 May 2006)

The following T10 approved proposal was incorporated in SAM-4 in this revision:

a) 06-116r3 - SAM-3: Converting to UML part 1; and
b) 06-198r0 - SAM-4, SPC-4: making the basic task management model obsolete.

R.8 Revision SAM-4r07 (24 July 2006)

The following T10 approved proposal was incorporated in SAM-4 in this revision:

a) 06-259r1 - SAM-4, et al.: making linked commands obsolete; and
b) 06-323r1 - SAM-4 SPC-4 et al Multiple service delivery subsystem editorial tweaks.

R.9 Revision SAM-4r08 (16 November 2006)

The following T10 approved proposal was incorporated in SAM-4 in this revision:

a) 06-411r2 - Clear REPORTED LUNS DATA HAS CHANGED on any command;
a) 06-282r4 - SPC-4 WRITE BUFFER clarifications; and
b) Removed the lingering target/initiator terminology which was emulated what the 06-116 document was accepted.

R.10 Revision SAM-4r09 (18 January 2007)

The following T10 approved proposal was incorporated in SAM-4 in this revision:

a) 06-451r6 - SAS-2 SAM-4: Miscellaneous State Machine Fixes.

Based on comments received in 06-341r1 several editorial changes were made in 5.1, 5.4, 7.1, and 7.12. The main change was to add it subclauses for each command and status SCSI transport protocol services and for each task management SCSI transport protocol service.

R.11 Revision SAM-4r10 (22 March 2007)

Based on the following 1/26/2007 E-mail from Rob Elliott changes were made as recommended in the e-mail.

1) In 5.4.2.4, "Send Command transport" s/b "Send Command Complete transport".
2) In 5.8.7 item a, add a space after the last comma
3) In 5.8.6 item c, the p should be capitalized
4) On page 82, the CAP WG concluded that ACA should not be cleared by power loss expected.
5) In 6.3.5 "clear task set" should be in all caps and cross reference 7.5

Based on the following 3/16/2007 E-mail from Michael Banther the term SCSI port name was removed from SAM-4. This deletion includes the last two paragraphs in section 4.5.5.2 and its glossary entry. This term should have been deleted when we switched to using SCSI target port name and SCSI initiator port name. The text of the E-mail follows:

SAM4r9 includes two paragraphs in 4.5.5.2, Relative port identifier, that defines a SCSI port name. Figure 18, SCSI Port class, does not show this attribute. However subsequent class diagrams and text define a SCSI target port name and a SCSI initiator port name. I suspect that the text defining SCSI port name should have been deleted but accidentally remains.

The following T10 approved proposals were incorporated in SAM-4 in this revision:

a) 07-105r2 (Comments from Adam Richter);
b) 07-066r1 (QUERY TASK SET task management function); and
c) 07-067r1 (QUERY UNIT ATTENTION task management function).

R.12 Revision SAM-4r11 (18 May 2007)

The following T10 approved proposal was incorporated in SAM-4 in this revision:

a) 07-224r0 - SAM-4 Include deferred errors in QUERY UNIT ATTENTION.

As a result of SAM-4 editing sessions the following changes were made:

a) Removed annex B;
b) Removed the following terms from the Definitions section as they are no longer used:
    A) device identifier;
    B) initiator;
    C) initiator identifier;
    D) port;
    E) SCSI device identifier;
    F) SCSI identifier;
    G) target; and
    H) target identifier;
c) Removed some cannots;
d) Removed all references to SPI-5;
e) Added ADT into annex A;
f) Added color to some of the figures;
g) Added color to ULM diagrams; and
h) Capitalized class names and attributes.

Contents

Page

Tables

Figures

Foreword

This foreword is not part of American National Standard INCITS.***:200x.

The purpose of this standard is to provide a basis for the coordination of SCSI standards development and to define requirements, common to all SCSI technologies and implementations, that are essential for compatibility with host SCSI application software and device-resident firmware across all SCSI transport protocols. These requirements are defined through a reference model that specifies the behavior and abstract structure that is generic to all SCSI I/O system implementations.

With any technical document there may arise questions of interpretation as new products are implemented. INCITS has established procedures to issue technical opinions concerning the standards developed by INCITS. These procedures may result in SCSI Technical Information Bulletins being published by INCITS.

These Bulletins, while reflecting the opinion of the Technical Committee that developed the standard, are intended solely as supplementary information to other users of the standard. This standard, ANSI INCITS.***:200x, as approved through the publication and voting procedures of the American National Standards Institute, is not altered by these bulletins. Any subsequent revision to this standard may or may not reflect the contents of these Technical Information Bulletins.

Current INCITS practice is to make Technical Information Bulletins available through:

INCITS Online Store	http://www.techstreet.com/INCITS.html
managed by Techstreet	Telephone: 1-734-302-7801 or
1327 Jones Drive	1-800-699-9277
Ann Arbor, MI 48105	Facsimile: 1-734-302-7811

or

Global Engineering	http://global.ihs.com/
15 Inverness Way East	Telephone: 1-303-792-2181 or
Englewood, CO 80112-5704	1-800-854-7179
	Facsimile: 1-303-792-2192

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the INCITS Secretariat, National Committee for Information Technology Standards, Information Technology Institute, 1250 Eye Street, NW, Suite 200, Washington, DC 20005- 3922.

This standard was processed and approved for submittal to ANSI by the InterNational Committee for Information Technology Standards (INCITS). Committee approval of the standard does not necessarily imply that all committee members voted for approval. At the time of it approved this standard, INCITS had the following members:

<<Insert INCITS member list>>

The INCITS Technical Committee T10 on SCSI Storage Interfaces, that reviewed this standard, had the following members:

Introduction

The SCSI Architecture Model - 4 standard is divided into the following clauses and annexes:

Clause 1 is the scope.
Clause 2 enumerates the normative references that apply to this standard.
Clause 3 describes the definitions, symbols, and abbreviations used in this standard.
Clause 4 describes the overall SCSI architectural model.
Clause 5 describes the SCSI command model element of the SCSI architecture.
Clause 6 describes the events that may be detected by a SCSI device.
Clause 7 describes the task management functions common to SCSI devices.
Clause 8 describes the task set management capabilities common to SCSI devices.
Annex A summarizes the identifier and name definitions of the SCSI transport protocols.

American National Standard
for Information Technology -

SCSI Architecture Model - 4 (SAM-4)

1 Scope

1.1 Introduction

The set of SCSI (Small Computer System Interface) standards consists of this standard and the SCSI implementation standards described in 1.3. This standard defines a reference model that specifies common behaviors for SCSI devices, and an abstract structure that is generic to all SCSI I/O system implementations.

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 following architecture model concepts from previous versions of this standard are made obsolete by this standard:

a) Support for the SPI-5 SCSI transport protocol;
b) Contingent Allegiance;
c) The TARGET RESET task management function;
d) Basic task management model;
e) Untagged tasks; and
f) Linked command function.

1.2 Requirements precedence

This standard defines generic requirements that pertain to SCSI implementation standards and implementation requirements. An implementation requirement specifies behavior in terms of measurable or observable parameters that apply to an implementation. Examples of implementation requirements defined in this document are the status values to be returned upon command completion and the service responses to be returned upon task management function completion.

Generic requirements are transformed to implementation requirements by an implementation standard. An example of a generic requirement is the hard reset behavior specified in 6.3.2.

.

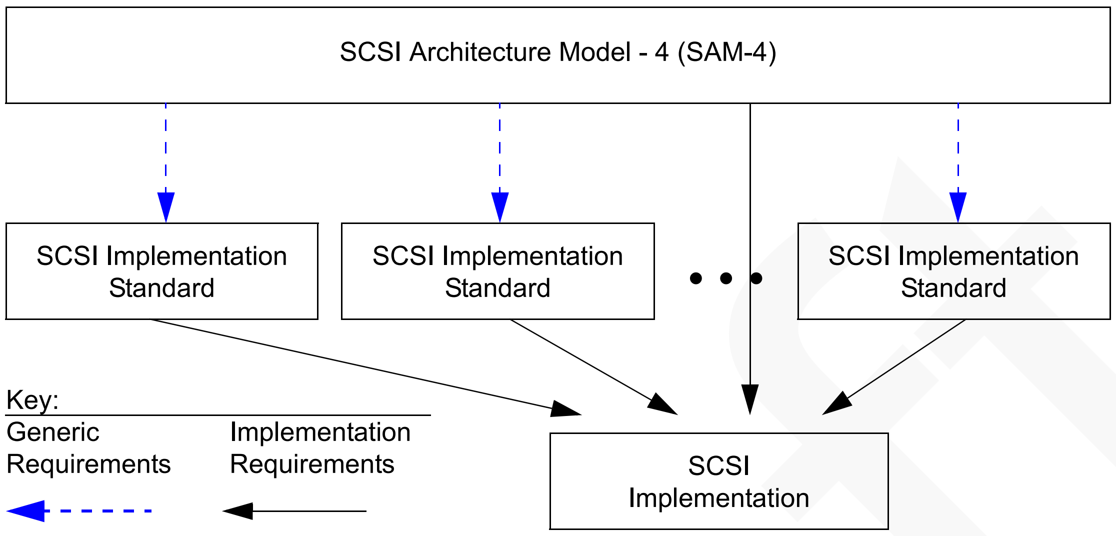

SCSI Architecture Model - 4 (SAM-4)
SCSI Implementation Standard
SCSI Implementation Standard
SCSI Implementation Standard
SCSI Implementation
Key:
Generic Requirements
Implementation Requirements

Figure 1 — Requirements precedence

As shown in figure 1, all SCSI implementation standards shall reflect the generic requirements defined herein. In addition, an implementation claiming SCSI compliance shall conform to the applicable implementation requirements defined in this standard and the appropriate SCSI implementation standards. In the event of a conflict between this document and other SCSI standards under the jurisdiction of technical committee T10, the requirements of this standard shall apply.

1.3 SCSI standards family

Figure 2 shows the relationship of this standard to the other standards and related projects in the SCSI family of standards as of the publication of this standard.

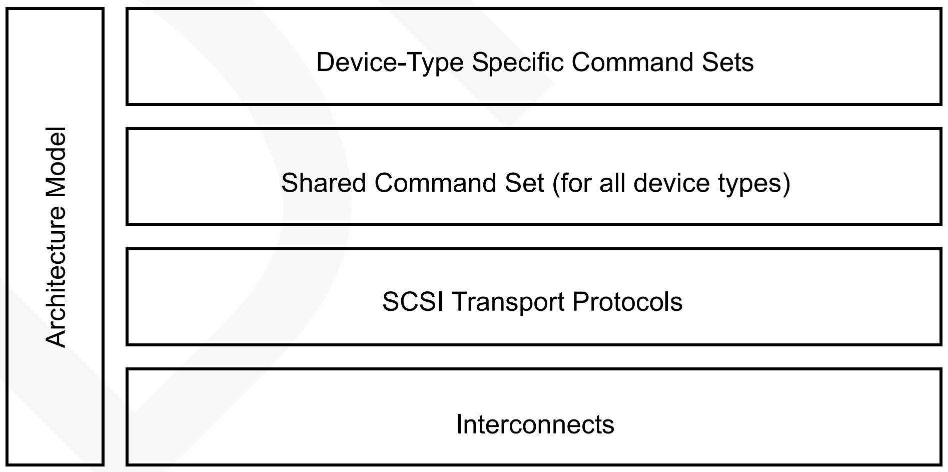

Architecture Model
Device-Type Specific Command Sets
Shared Command Set (for all device types)
SCSI Transport Protocols
Interconnects

Figure 2 — SCSI document structure

The roadmap in figure 2 is intended to show the general applicability of the documents to one another. Figure 2 is not intended to imply a relationship such as a hierarchy, protocol stack, or system architecture.

The functional areas identified in figure 2 characterize the scope of standards within a group as follows:

Architecture Model: Defines the SCSI systems model, the functional partitioning of the SCSI standard set and requirements applicable to all SCSI implementations and implementation standards.

Device-Type Specific Command Sets: Implementation standards that define specific device types including a device model for each device type. These standards specify the required commands and behavior that is specific to a given device type and prescribe the requirements to be followed by a SCSI initiator device when sending commands to a SCSI target device having the specific device type. The commands and behaviors for a specific device type may include by reference commands and behaviors that are shared by all SCSI devices.

Shared Command Set: An implementation standard that defines a model for all SCSI device types. This standard specifies the required commands and behavior that is common to all SCSI devices, regardless of device type, and prescribes the requirements to be followed by a SCSI initiator device when sending commands to any SCSI target device.

SCSI Transport Protocols: Implementation standards that define the requirements for exchanging information so that different SCSI devices are capable of communicating.

Interconnects: Implementation standards that define the communications mechanism employed by the SCSI transport protocols. These standards may describe the electrical and signaling requirements essential for SCSI devices to interoperate over a given interconnect. Interconnect standards may allow the interconnection of devices other than SCSI devices in ways that are outside the scope of this standard.

The term SCSI is used to refer to the family of standards described in this subclause.

2 Normative references

2.1 Normative references

The following standards contain provisions that, by reference in the text, constitute provisions of this standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below.

Copies of the following documents may be obtained from ANSI: approved ANSI standards, approved and draft international and regional standards (ISO, IEC, CEN/CENELEC, ITUT), and approved and draft foreign standards (including BSI, JIS, and DIN). For further information, contact ANSI Customer Service Department at 212-642-4900 (phone), 212-302-1286 (fax) or via the World Wide Web at http://www.ansi.org.

2.2 Approved references

ISO/IEC 14776-452, *SCSI Primary Commands - 2 (SPC-2)* [ANSI NCITS.351-2001]

ISO/IEC 60027-2-am2, (1999-01) *Letter symbols to be used in electrical technology - Part 2: Telecommunications and electronics (Amendment 2)*

2.3 References under development

At the time of publication, the following referenced standards were still under development. For information on the current status of the document, or regarding availability, contact the relevant standards body or other organization as indicated.

ISO/IEC 14776-453, *SCSI Primary Commands - 3 (SPC-3)* [T10/1416-D]

ISO/IEC 14776-322, *SCSI Block Commands - 2 (SBC-2)* [T10/1417-D]

Editor's Note 1: Need to look into all the references for this standard.

3 Definitions, symbols, abbreviations, and conventions

3.1 Definitions

3.1.1 ACA command: A command performed by a task with the ACA 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 used in class diagrams, a form of association that defines a whole-part relationship between the whole (i.e., aggregate) and its parts.

3.1.4 application client: An object that is the source of commands and task management function requests.

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 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).

3.1.7 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.

3.1.8 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 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).

3.1.26 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: An object within a logical unit that processes SCSI tasks according to the requirements for task management described in clause 8.

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 used in class diagrams, 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 device name: A SCSI device name of a SCSI initiator device. See 4.5.9 .

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

3.1.57 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

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

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 object, containing a device server and task manager, 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 used in class diagrams, 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 (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 question. A request may change the state of the object but a 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 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.

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

3.1.92 SCSI device: A device that contains one or more SCSI ports that are connected to a service delivery subsystem and supports a SCSI application protocol.

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 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.

3.1.97 SCSI initiator port: A 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.

3.1.98 SCSI port: A 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).

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 3.1.56) or a target port identifier (see 3.1.117).

3.1.100 SCSI target device: A 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.

3.1.101 SCSI target port: A 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.

3.1.102 SCSI transport protocol layer (STPL): The protocol and services used by a SCSI application layer 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: 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.

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 device name: A SCSI device name (see 3.1.93) of a SCSI target device. See 4.5.14 .

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

3.1.118 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 .

3.1.119 task: An object within the logical unit representing the work associated with a command. See 4.5.19 .

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

3.1.121 task tag: An attribute of a task class containing up to 64 bits 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 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: An object in a SCSI target port that routes commands and task management functions between a service delivery subsystem (see 3.1.113) and the appropriate task manager(s) (see 4.5.8).

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

3.1.128 **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.

3.1.129 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.

3.1.130 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 .

3.1.131 well known logical unit: 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 .

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

3.2 Acronyms

ACA	Auto Contingent Allegiance (see 3.1.8)
CDB	Command Descriptor Block (see 3.1.19)
CRN	Command Reference Number
ISO	Organization for International Standards
LUN	Logical Unit Number (see 3.1.66)
MMC-2	SCSI Multi-Media Commands -2 (see 1.3)
n/a	Not Applicable
RAID	Redundant Array of Independent Disks
SAL	SCSI application layer (see 3.1.91)
SBC	SCSI-3 Block Commands (see 1.3)
SCSI	The architecture defined by the family of standards described in 1.3
SPC-3	SCSI Primary Commands -3 (see 1.3)
STPL	SCSI transport protocol layer (see 3.1.102)
SSC	SCSI-3 Stream Commands (see 1.3)
VPD	Vital Product Data (see SPC-3)
W-LUN	Well known logical unit number (see 3.1.132)

3.3 Keywords

3.3.1 invalid: A keyword used to describe an illegal or unsupported bit, byte, word, field or code value. Receipt by a device server of an invalid bit, byte, word, field or code value shall be reported as error.

3.3.2 mandatory: A keyword indicating an item that is required to be implemented as defined in this standard.

3.3.3 may: A keyword that indicates flexibility of choice with no implied preference (synonymous with may or may not).

3.3.4 may not: A keyword that indicates flexibility of choice with no implied preference (synonymous with may or may not).

3.3.5 obsolete: A keyword indicating that an item was defined in prior SCSI standards but has been removed from this standard.

3.3.6 option, optional: Keywords that describe features that are not required to be implemented by this standard. However, if any optional feature defined by this standard is implemented, then it shall be implemented as defined in this standard.

3.3.7 SCSI transport protocol specific: Implementation of the referenced item is defined by a SCSI transport protocol standard (see 1.3).

3.3.8 reserved: A keyword referring to bits, bytes, words, fields, and code values that are set aside for future standardization. A reserved bit, byte, word, or field shall be set to zero, or in accordance with a future extension to this standard. Recipients are not required to check reserved bits, bytes, words, or fields for zero values. Receipt of reserved code values in defined fields shall be reported as error.

3.3.9 shall: A keyword indicating a mandatory requirement. Designers are required to implement all such mandatory requirements to ensure interoperability with other products that conform to this standard.

3.3.10 should: A keyword indicating flexibility of choice with a strongly preferred alternative; equivalent to the phrase "it is strongly recommended".

3.3.11 vendor specific: Specification of the referenced item is determined by the SCSI device vendor.

3.4 Editorial conventions

Certain words and terms used in this standard have a specific meaning beyond the normal English meaning. These words and terms are defined either in the glossary or in the text where they first appear.

Upper case is used when referring to the name of a numeric value defined in this specification or a formal attribute possessed by an entity. When necessary for clarity, names of objects, procedure calls, arguments or discrete states are capitalized or set in bold type. Names of fields are identified using small capital letters (e.g., NACA bit).

Names of procedure calls are identified by a name in bold type, such as **Execute Command** (see clause 5). Names of arguments are denoted by capitalizing each word in the name. For instance, Sense Data is the name of an argument in the **Execute Command** procedure call.

Quantities having a defined numeric value are identified by large capital letters. CHECK CONDITION, for example, refers to the numeric quantity defined in table 25 (see 5.3.1). Quantities having a discrete but unspecified value are identified using small capital letters. As an example, TASK COMPLETE, indicates a quantity returned by the **Execute Command** procedure call (see clause 5). Such quantities are associated with an event or indication whose observable behavior or value is specific to a given implementation standard.

Lists sequenced by letters (e.g., a-red, b-blue, c-green) show no priority relationship between the listed items. Numbered lists (e.g., 1-red, 2-blue, 3-green) show a priority ordering between the listed items.

If a conflict arises between text, tables, or figures, the order of precedence to resolve the conflicts is text; then tables; and finally figures. Not all tables or figures are fully described in the text. Tables show data format and values.

Notes do not constitute any requirements for implementors.

3.5 Numeric conventions

A binary number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 and 1 immediately followed by a lower-case b (e.g., 0101b). Underscores or spaces may be included in binary number representations to increase readability or delineate field boundaries (e.g., 0 0101 1010b or 0_0101_1010b).

A hexadecimal number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 through 9 and/or the upper-case English letters A through F immediately followed by a lower-case h (e.g., FA23h). Underscores or spaces may be included in hexadecimal number representations to increase readability or delineate field boundaries (e.g., B FD8C FA23h or B_FD8C_FA23h).

A decimal number is represented in this standard by any sequence of digits consisting of only the Western-Arabic numerals 0 through 9 not immediately followed by a lower-case b or lower-case h (e.g., 25).

This standard uses the ISO convention for representing decimal numbers (e.g., the thousands and higher multiples are separated by a space and a comma is used as the decimal point). Table 1 shows some examples of decimal numbers represented using the ISO and American conventions.

Table 1 — ISO and American numbering conventions examples

ISO	American
0,6	0.6
3,141 592 65	3.14159265
1 000	1,000
1 323 462,95	1,323,462.95

3.6 Notation conventions

3.6.1 Notation conventions overview

This standard uses class diagrams and object diagrams with notation that is based on the Unified Modeling Language (UML).

See 3.6.2 for the conventions used for class diagrams.

See 3.6.3 for the conventions used for object diagrams.

Within class diagrams and object diagrams there may be constraints which specify requirements and notes which are informative.

A constraint is specified as text encapsulated with a { } notation within a box. See figure 5 for an example of a constraint.

A note is specified as text within a box (i.e., no { }). See figure 6 for an example of a note.

3.6.2 Class diagram conventions

Figure 3 shows the notation used for classes in class diagrams.

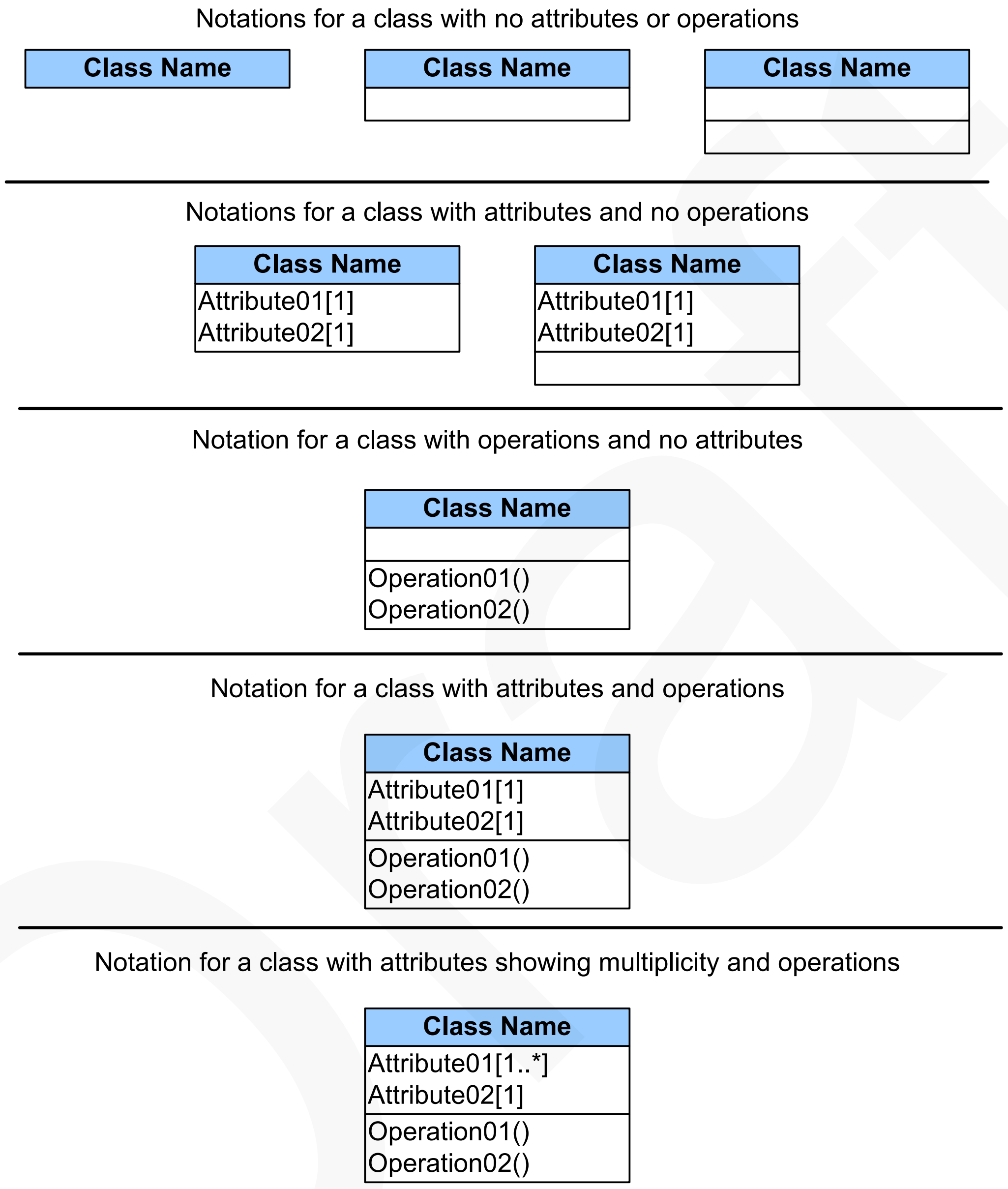

Notations for a class with no attributes or operations
Class Name
Class Name
Class Name
Notations for a class with attributes and no operations
Class Name
Attribute01[1]
Attribute02[1]
Class Name
Attribute01[1]
Attribute02[1]
Notation for a class with operations and no attributes
Class Name
Operation01()
Operation02()
Notation for a class with attributes and operations
Class Name
Attribute01[1]
Attribute02[1]
Operation01()
Operation02()
Notation for a class with attributes showing multiplicity and operations
Class Name
Attribute01[1..*]
Attribute02[1]
Operation01()
Operation02()

Figure 3 — Class diagram conventions

See table 2 for the notation used to indicate multiplicity.

Table 2 — 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 4 and figure 5 for examples of multiplicity notation.

Solid lines with arrowheads (see figure 4) are the notation used to describe the association relationship between classes in class diagrams. Multiplicity notation occurs at each end of the solid line.

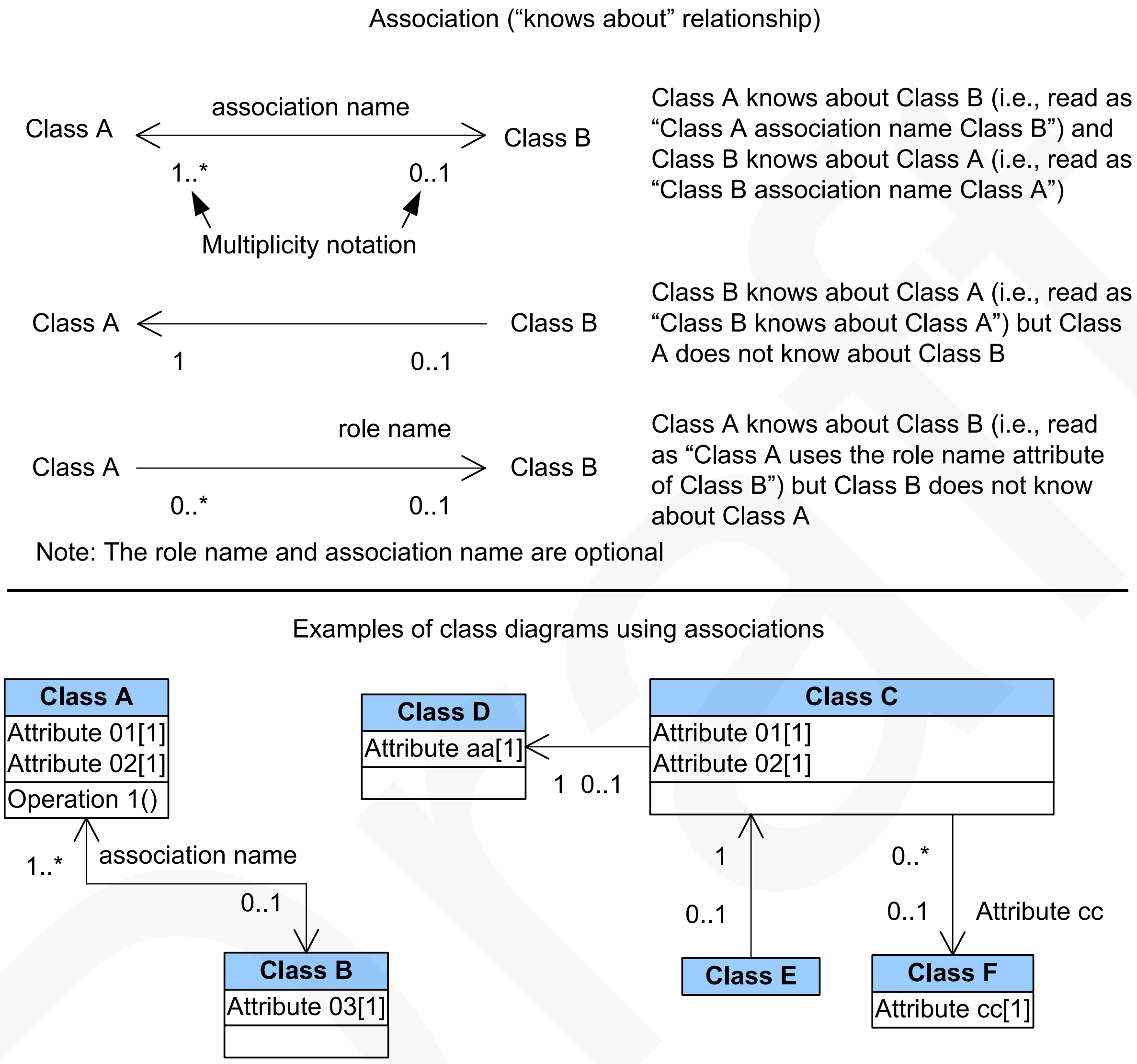

Association (“knows about” relationship)
Class A
association name
Class B
1..*
0..1
Multiplicity notation
Class A knows about Class B (i.e., read as “Class A association name Class B”) and Class B knows about Class A (i.e., read as “Class B association name Class A”)
Class A
Class B
1
0..1
Class B knows about Class A (i.e., read as “Class B knows about Class A”) but Class A does not know about Class B
Class A
role name
Class B
0..*
0..1
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
Class A
Attribute 01[1]
Attribute 02[1]
Operation 1()
1..*
association name
0..1
Class B
Attribute 03[1]
Class D
Attribute aa[1]
1 0..1
Class C
Attribute 01[1]
Attribute 02[1]
1
0..1
Class E
0..*
0..1
Attribute cc
Class F
Attribute cc[1]

Figure 4 — Notation for association relationships for class diagrams

Solid lines with diamonds (see figure 5) are the notation used to describe the aggregation relationship between classes in class diagrams. Multiplicity notation occurs at each end of the solid line.

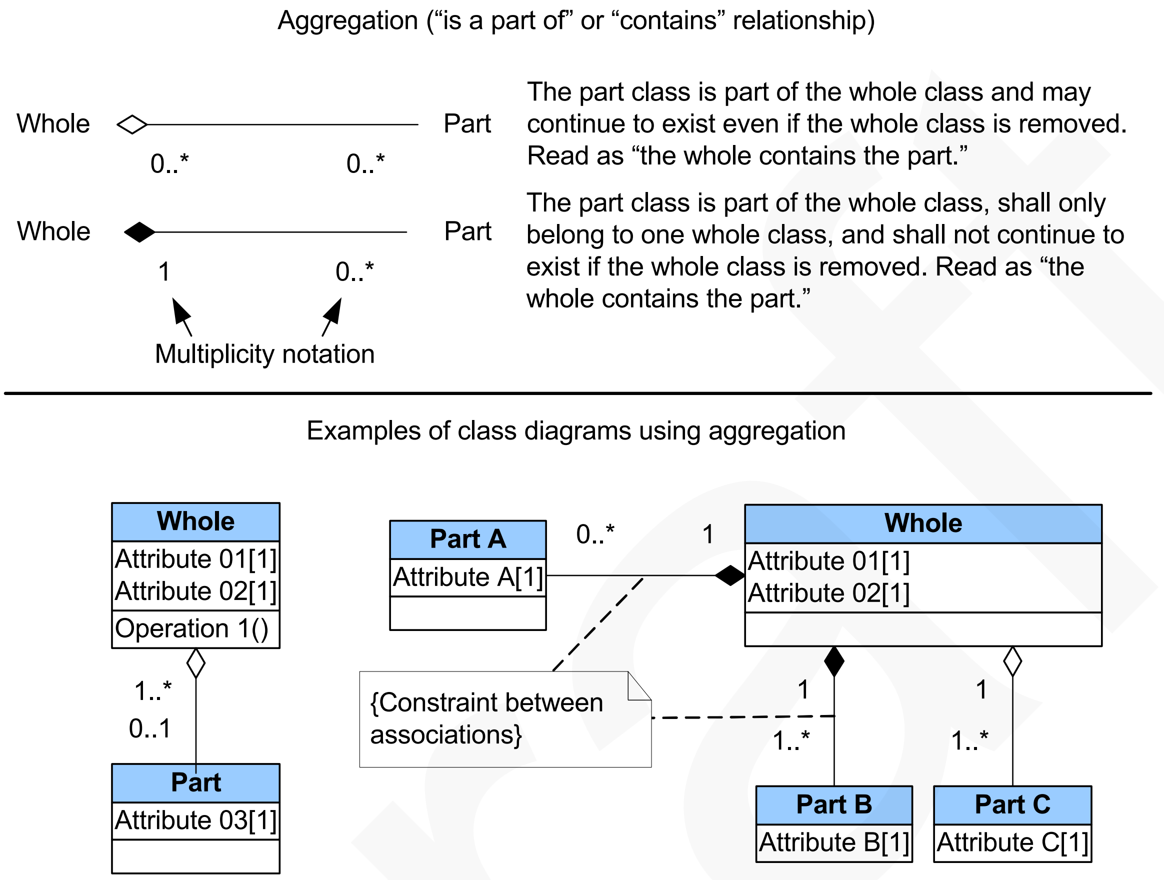

Aggregation ("is a part of" or "contains" relationship)
Whole
0..*
0..*
Part
The part class is part of the whole class and may continue to exist even if the whole class is removed. Read as "the whole contains the part."
Whole
1
0..*
Part
The part class is part of the whole class, shall only belong to one whole class, and shall not continue to exist if the whole class is removed. Read as "the whole contains the part."
Multiplicity notation
Examples of class diagrams using aggregation
Whole
Attribute 01[1]
Attribute 02[1]
Operation 1()
1..*
0..1
Part
Attribute 03[1]
Part A
Attribute A[1]
0..*
1
Whole
Attribute 01[1]
Attribute 02[1]
{Constraint between associations}
1
1..*
1
1..*
Part B
Attribute B[1]
Part C
Attribute C[1]

Figure 5 — Notation for aggregation relationships for class diagrams

Solid lines with triangles (see figure 6) are the notation used to describe the generalization relationship between classes in class diagrams.

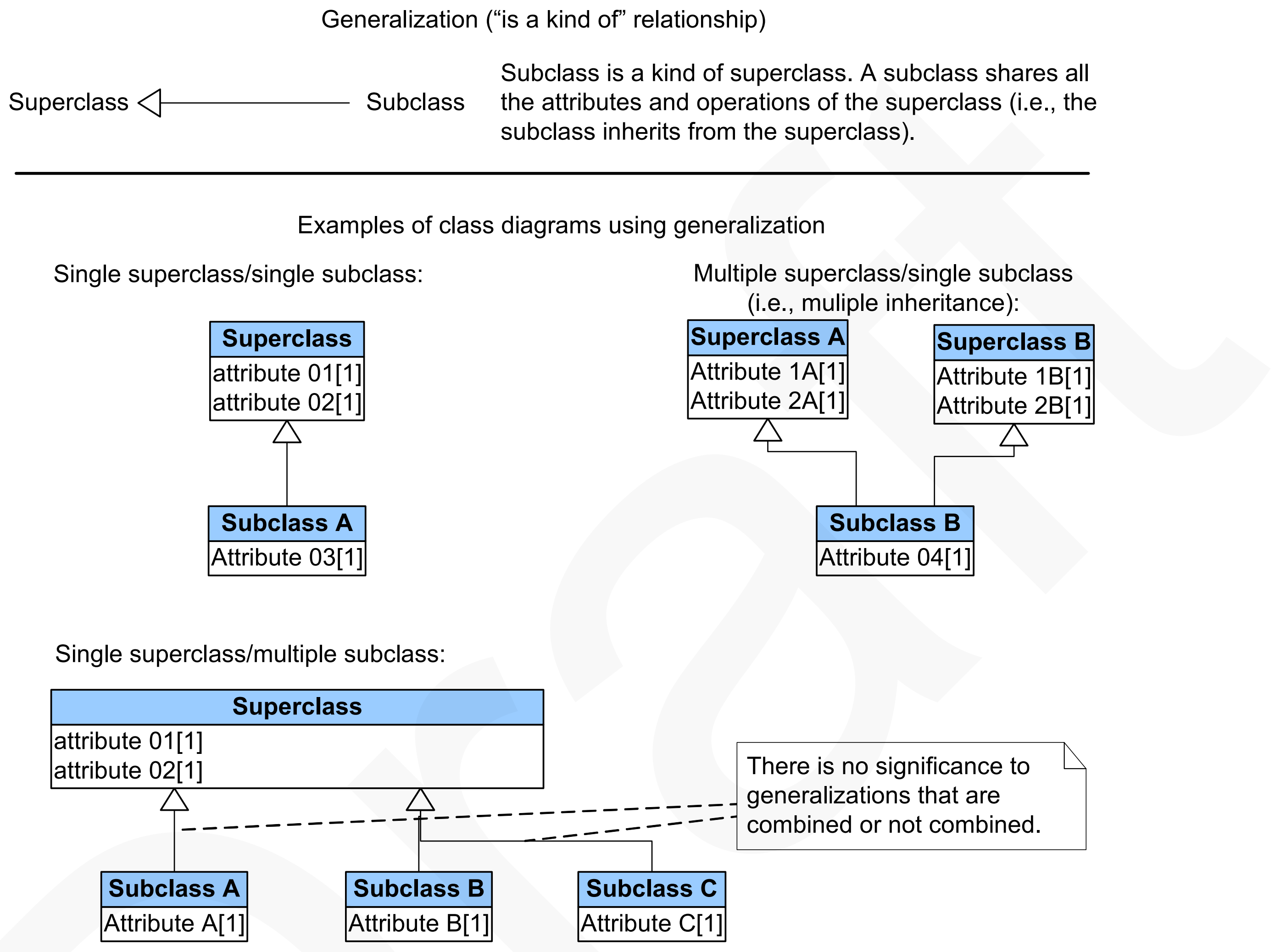

Generalization (“is a kind of” relationship)
Superclass
Subclass
Subclass is a kind of superclass. A subclass shares all the attributes and operations of the superclass (i.e., the subclass inherits from the superclass).
Examples of class diagrams using generalization
Single superclass/single subclass:
Superclass
attribute 01[1]
attribute 02[1]
Subclass A
Attribute 03[1]
Multiple superclass/single subclass (i.e., muliple inheritance):
Superclass A
Attribute 1A[1]
Attribute 2A[1]
Superclass B
Attribute 1B[1]
Attribute 2B[1]
Subclass B
Attribute 04[1]
Single superclass/multiple subclass:
Superclass
attribute 01[1]
attribute 02[1]
There is no significance to generalizations that are combined or not combined.
Subclass A
Attribute A[1]
Subclass B
Attribute B[1]
Subclass C
Attribute C[1]

Figure 6 — Notation for generalization relationships for class diagrams

Dashed lines with arrowheads (see figure 7) are the notation used to describe the dependency relationship between classes in class diagrams.

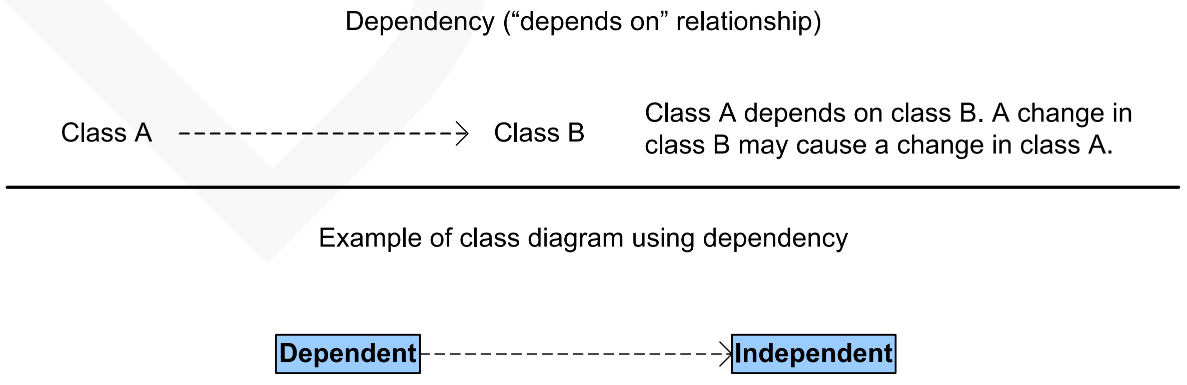

Dependency (“depends on” relationship)
Class A
Class B
Class A depends on class B. A change in class B may cause a change in class A.
Example of class diagram using dependency
Dependent
Independent

Figure 7 — Notation for dependency relationships for class diagrams

3.6.3 Object diagram conventions

Figure 8 shows the notation used for objects in object diagrams.

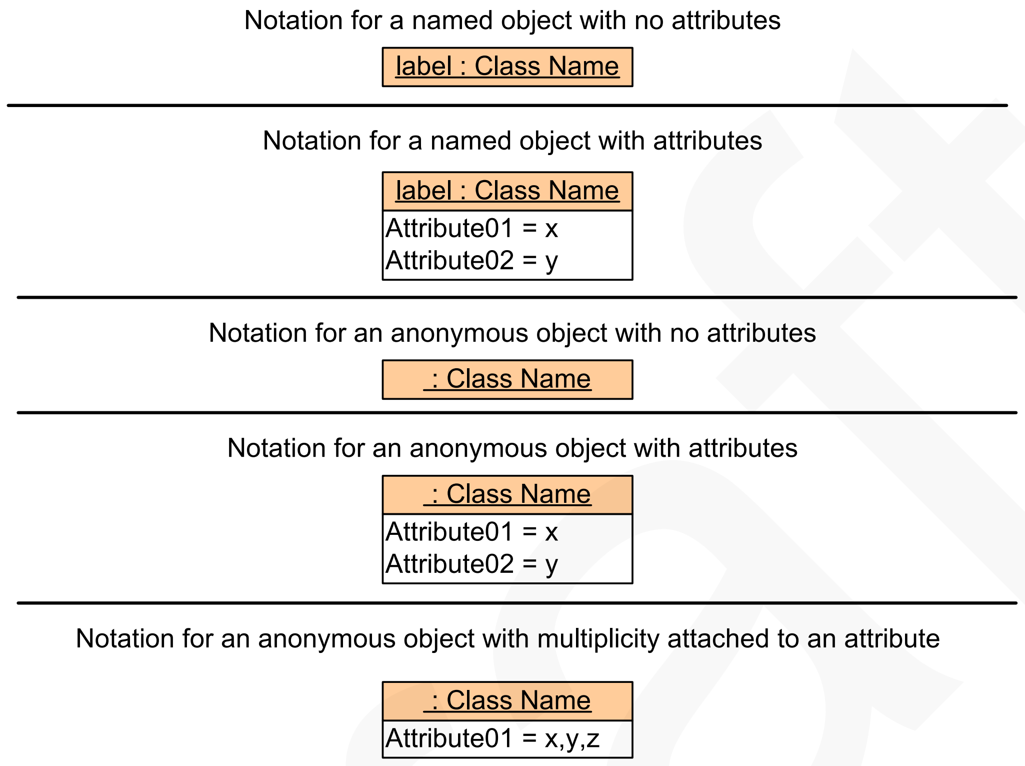

Notation for a named object with no attributes
label : Class Name
Notation for a named object with attributes
label : Class Name
Attribute01 = x
Attribute02 = y
Notation for an anonymous object with no attributes
: Class Name
Notation for an anonymous object with attributes
: Class Name
Attribute01 = x
Attribute02 = y
Notation for an anonymous object with multiplicity attached to an attribute
: Class Name
Attribute01 = x,y,z

Figure 8 — Object diagram conventions

Solid lines (see figure 6) are the notation used to describe the link relationship between objects in object diagrams.

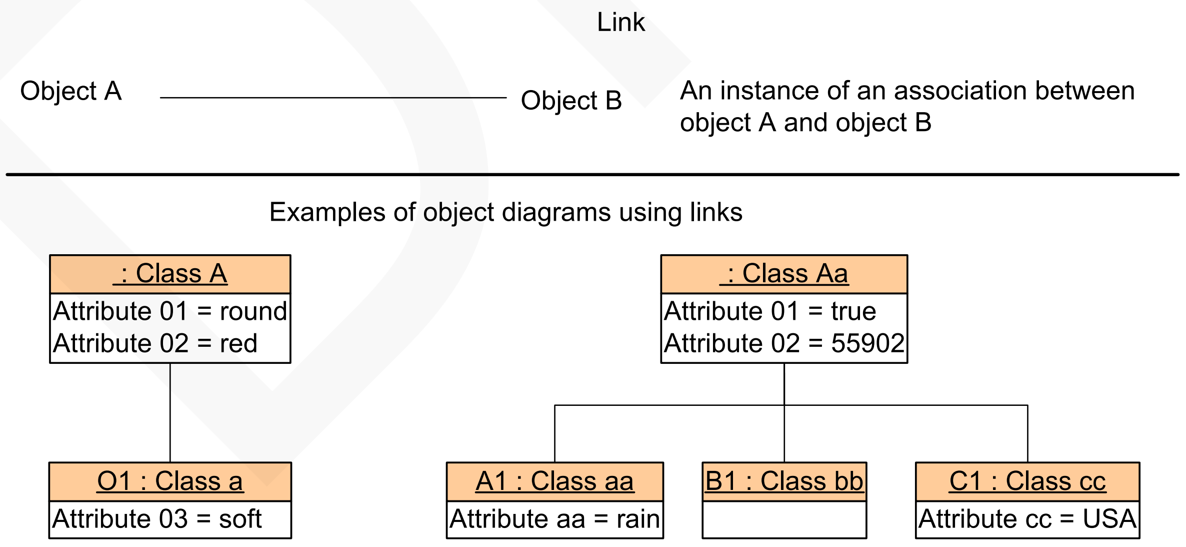


Figure 9 — Notation for link relationships for object diagrams

3.6.4 Notation for procedure calls

In this standard, the model for functional interfaces between entities is a procedure call (see 3.1.80). Such interfaces are specified using the following notation:

[Result =] Procedure Name (IN ([input-1] [,input-2] …]), OUT ([output-1] [,output-2] …))

Where:

Result:	A single value representing the outcome of the procedure call.
Procedure Name:	A descriptive name for the function modeled by the procedure call.
Input-1, Input-2, …:	A comma-separated list of names identifying caller-supplied input arguments.
Output-1, Output-2, …:	A comma-separated list of names identifying output arguments to be returned by the procedure call.
"[…]":	Brackets enclosing optional or conditional arguments.

This notation allows arguments to be specified as inputs and outputs. The following is an example of a procedure call specification:

Found = **Search** (IN (Pattern, Item List), OUT ([Item Found]))

Where:

Found = Flag

Flag, if set to one, indicates that a matching item was located.

Input Arguments:

Pattern = … /* Definition of **Pattern** argument */

Argument containing the search pattern.

Item List = Item<NN> /* Definition of **Item List** as an array of NN **Item** arguments*/

Contains the items to be searched for a match.

Output Arguments:

Item Found = Item … /* Item located by the search procedure call */

This argument is only returned if the search succeeds.

3.6.5 Notation for state diagrams

All state diagrams use the notation shown in figure 10.

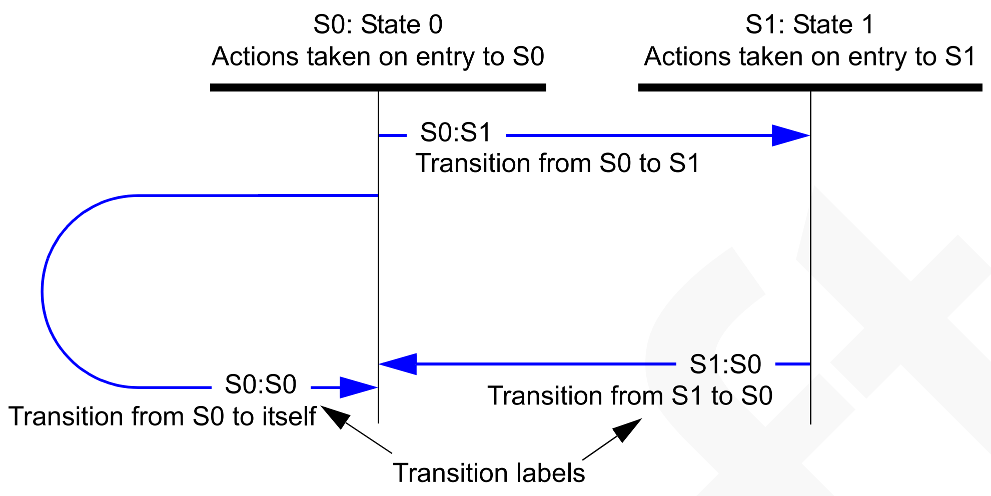


Figure 10 — Example state diagram

The state diagram is followed by a list of the state transitions, using the transition labels. Each transition is described in the list with particular attention to the conditions that cause the transition to occur and special conditions related to the transition. Using figure 10 as an example, the transition list might read as follows:

Transition S0:S1: This transition occurs when state S0 is exited and state S1 is entered.

Transition S1:S0: This transition occurs when state S1 is exited and state S0 is entered.

Transition S0:S0: This transition occurs when state S0 transitions to itself. The reason for a transition from S0 to itself is to specify that the actions taken whenever state S0 is entered are repeated every time the transition occurs.

A system specified in this manner has the following properties:

a) Time elapses only within discrete states;
b) State transitions are logically instantaneous; and
c) Every time a state is entered, the actions of that state are started. Note that this means that a transition that points back to the same state restarts the actions from the beginning.

4 SCSI architecture model

4.1 Introduction

The purpose of the SCSI architecture model is to:

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

The development of this standard is assisted by the use of an abstract model. To specify the external behavior of a SCSI system, elements in a system are replaced by functionally equivalent components within this model. Only externally observable behavior is retained as the standard of behavior. The description of internal behavior in this standard is provided only to support the definition of the observable aspects of the model. Those aspects are limited to the generic properties and characteristics needed for host applications to interoperate with SCSI devices in any SCSI interconnect and SCSI transport protocol environment. The model does not address other requirements that may be essential to some I/O system implementations (e.g., the mapping from SCSI device addresses to network addresses, the procedure for discovering SCSI devices on a network, and the definition of network authentication policies for SCSI initiator devices or SCSI target devices). These considerations are outside the scope of this standard.

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

The SCSI architecture model is described in terms of classes (see 3.1.13), protocol layers, and service interfaces between classes. As used in this standard, classes are abstractions, encapsulating a set of related functions (i.e., attributes), operations, data types, and other classes. Certain classes are defined by SCSI (e.g., an interconnect), while others are needed to understand the functioning of SCSI but have implementation definitions outside the scope of SCSI (e.g., a task). These classes exhibit well-defined and observable behaviors, but they do not exist as separate physical elements. A class may contain a single attribute (e.g., a task tag) or 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 7.12. 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 11. 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 a 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.

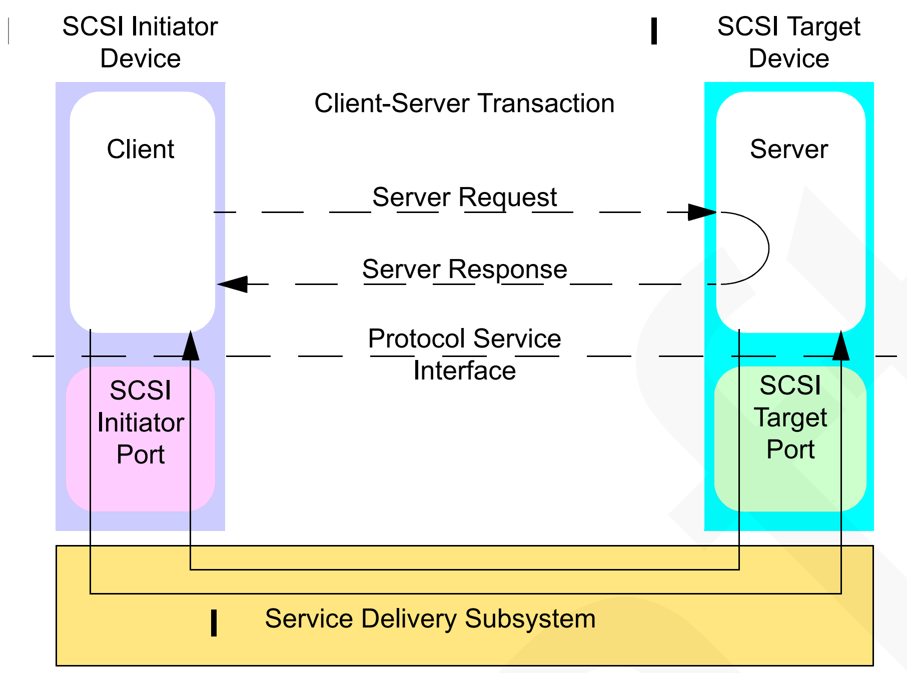


Figure 11 — Client-Server model

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 7.11).

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 a service delivery subsystem.

4.3 The SCSI client-server model

4.3.1 SCSI client-server model overview

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

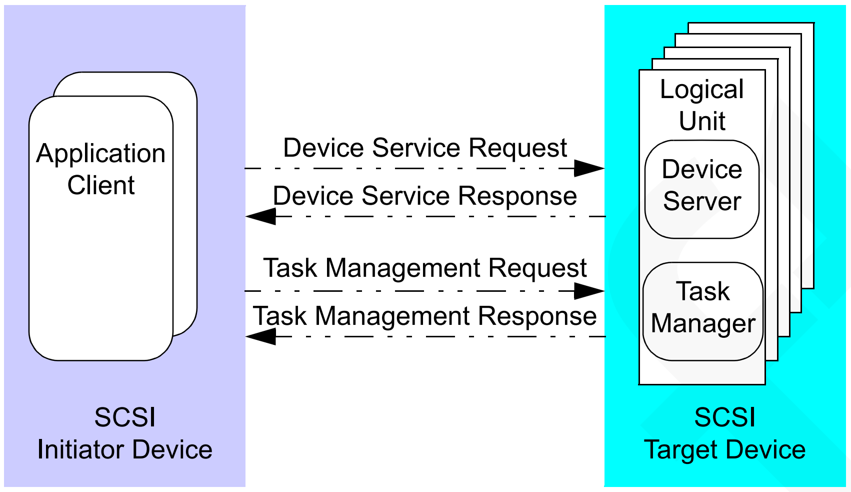


Figure 12 — SCSI client-server model

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

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.

4.3.2 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 a 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 7.12 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 a 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 a 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 in a SCSI I/O system as seen by the application clients interacting with the system. As shown in figure 13, 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.

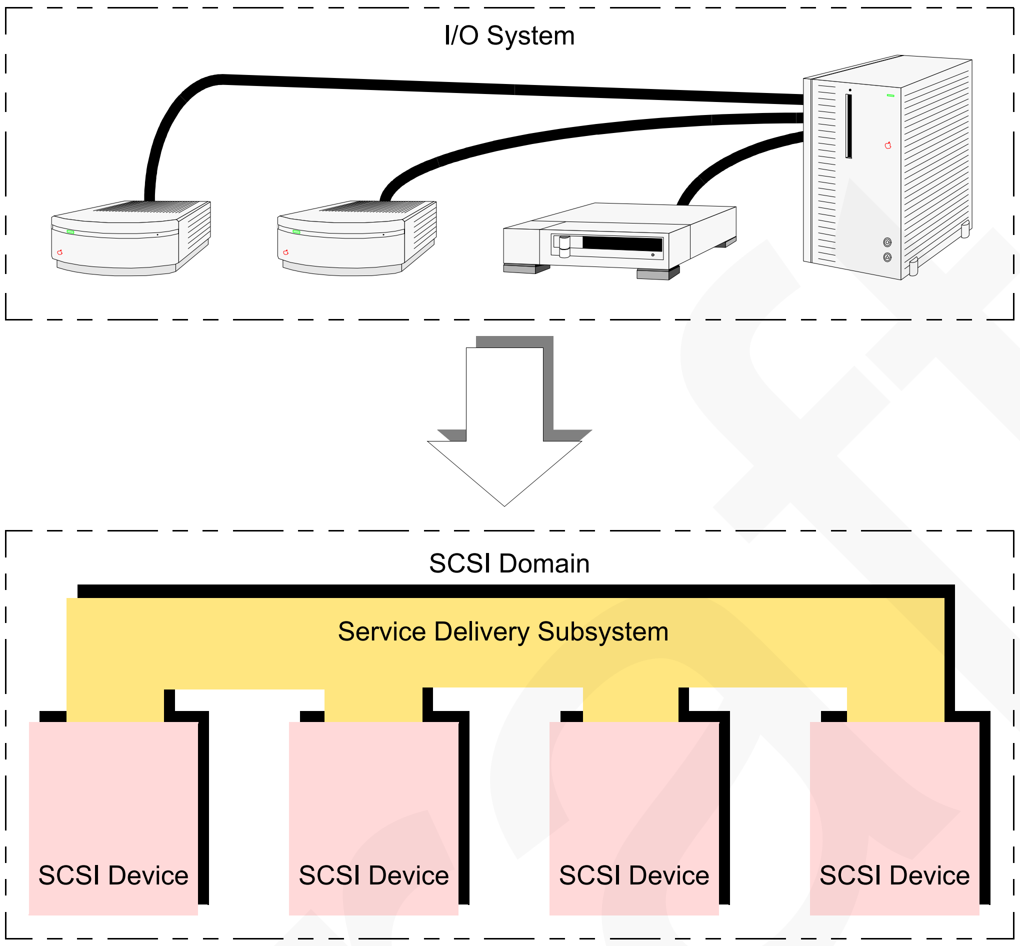


Figure 13 — SCSI I/O system and domain model

4.5 SCSI classes

4.5.1 SCSI classes overview

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

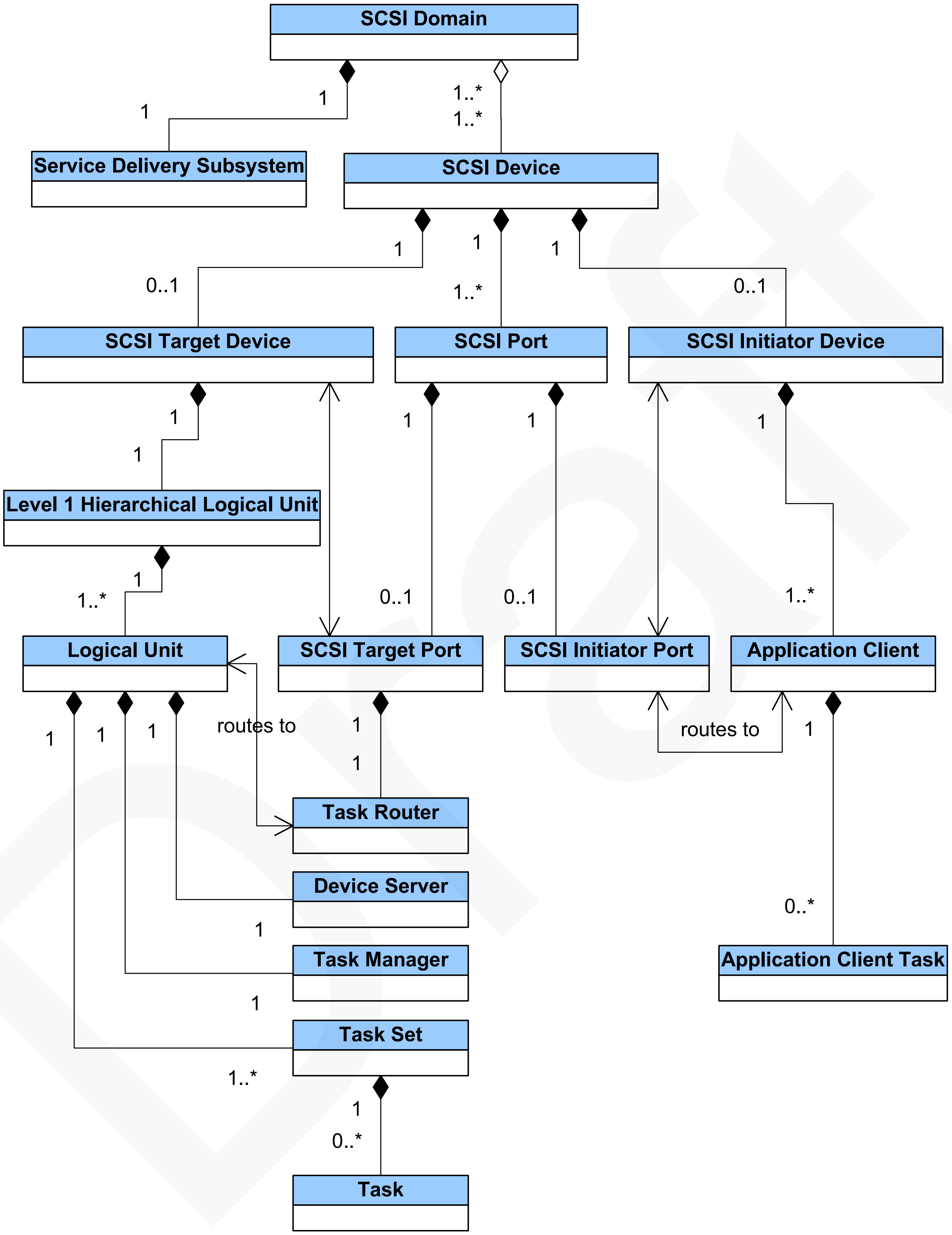


Figure 14 — SCSI Domain class diagram overview

4.5.2 SCSI Domain class

A SCSI Domain class (figure 15) contains the:

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

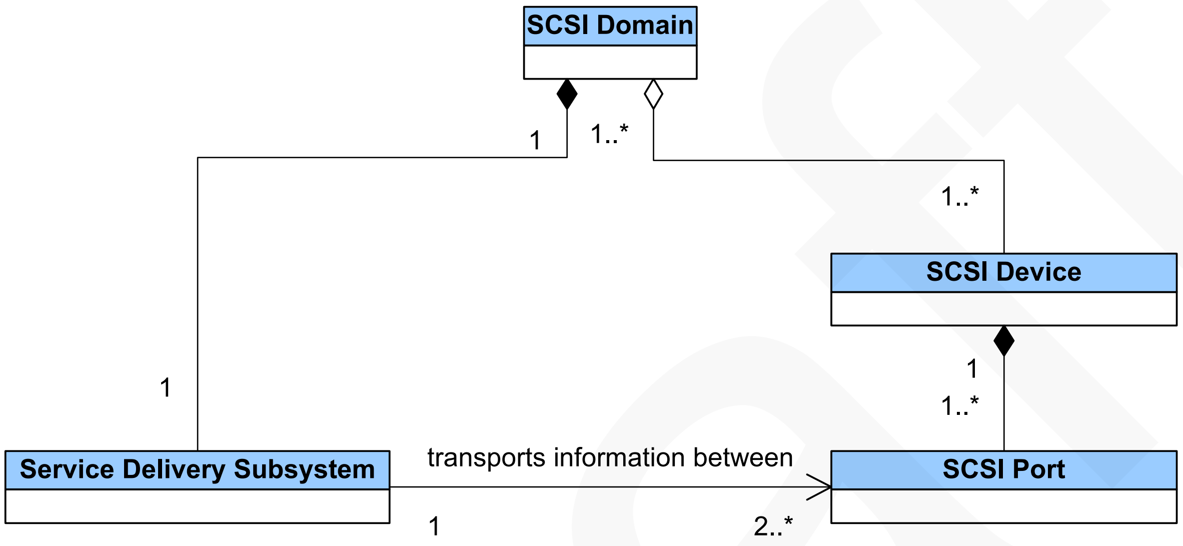


Figure 15 — SCSI Domain class diagram

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

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

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

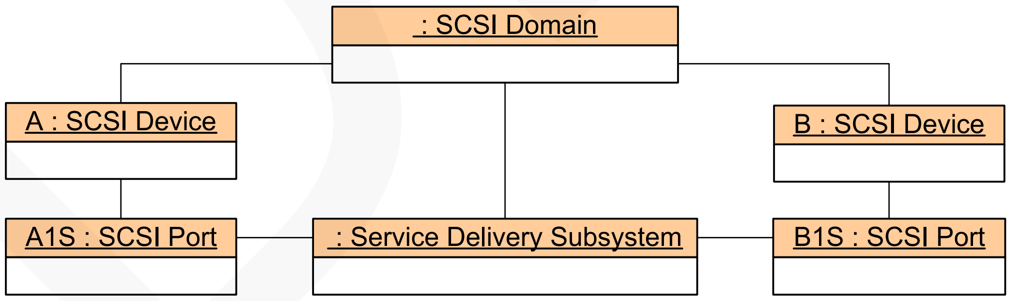


Figure 16 — SCSI domain object diagram

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 3.1.98) in the SCSI domain, providing a mechanism through which application clients communicate with device servers and task managers (see 4.5.3).

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

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

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

4.5.4 SCSI Device class

4.5.4.1 SCSI Device class overview

See figure 17 for the SCSI Device class diagram.

A 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 4.5.14), or both.

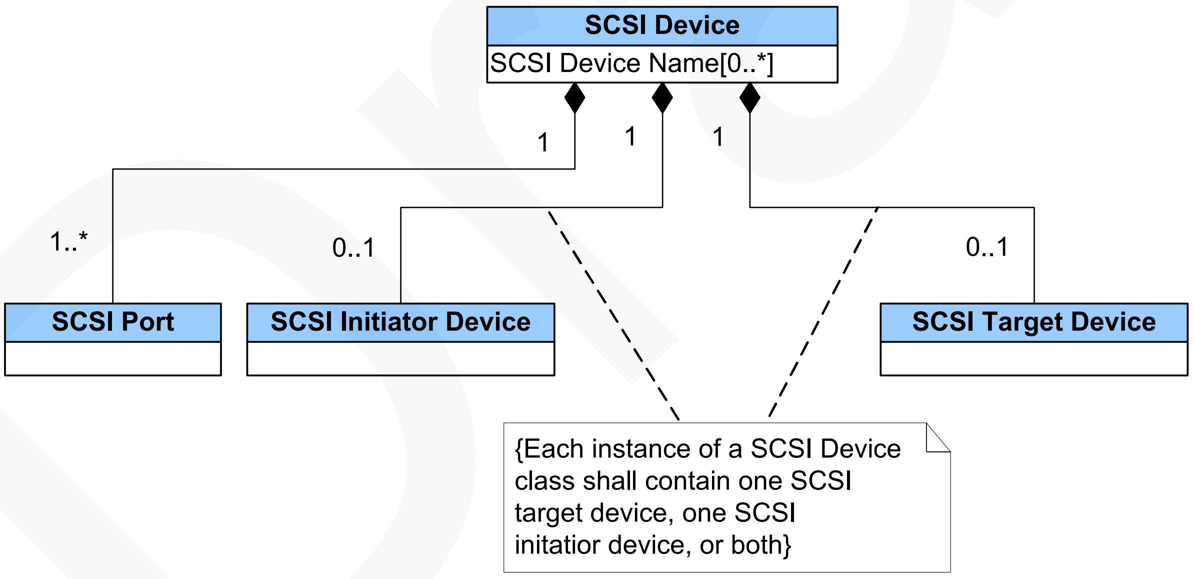

SCSI Device
SCSI Device Name[0..*]
1
1
1
1..*
0..1
0..1
SCSI Port
SCSI Initiator Device
SCSI Target Device
{Each instance of a SCSI Device class shall contain one SCSI target device, one SCSI initator device, or both}

Figure 17 — 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 3.1.68) for a SCSI device that is world wide unique within the SCSI transport protocol of each SCSI domain in which the SCSI device has SCSI ports. For each supported

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

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

4.5.5 SCSI Port class

4.5.5.1 SCSI Port class overview

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

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

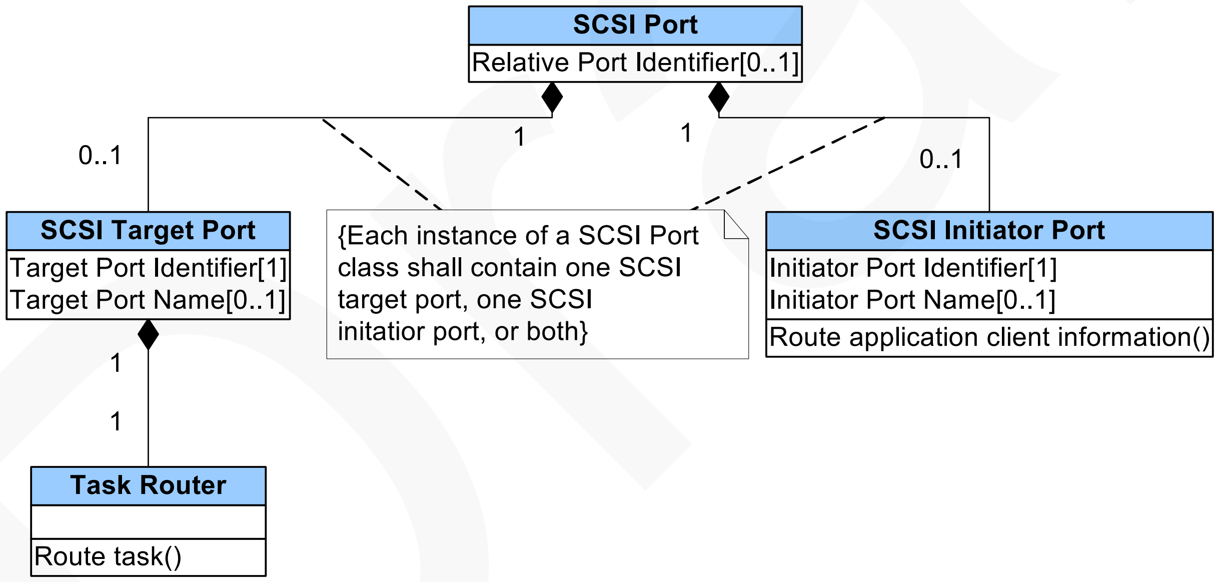

SCSI Port
Relative Port Identifier[0..1]
0..1
1
1
0..1
SCSI Target Port
Target Port Identifier[1]
Target Port Name[0..1]
{Each instance of a SCSI Port class shall contain one SCSI target port, one SCSI initatior port, or both}
SCSI Initiator Port
Initiator Port Identifier[1]
Initiator Port Name[0..1]
Route application client information()
1
1
Task Router
Route task()

Figure 18 — 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.

4.5.6 SCSI Target Port class

4.5.6.1 SCSI Target Port class overview

A SCSI Target Port class (see figure 18) contains the:

a) Task Router class (see 4.5.8);

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

4.5.6.2 TargeT Port Identifier attribute

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

4.5.6.3 Target Port Name attribute

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

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

4.5.7 SCSI Initiator Port class

4.5.7.1 SCSI Initiator Port class overview

The SCSI Initiator Port class:

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

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 3.1.68) of a SCSI initiator port that is world wide unique within the SCSI transport protocol of the SCSI domain of that SCSI initiator port. 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 Task Router class

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

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

a) Commands addressed to a valid logical unit are routed to the task manager in the specified logical unit;
b) Commands addressed to an incorrect logical unit are handled as described in 5.8.4;
c) Task management functions with I_T_L nexus scope (e.g., ABORT TASK SET, CLEAR TASK SET, CLEAR ACA, LOGICAL UNIT RESET, QUERY TASK SET, and QUERY UNIT ATTENTION) or I_T_L_Q nexus scope (e.g., ABORT TASK and QUERY TASK) addressed to a valid logical unit are routed to the task manager in the specified logical unit;
d) Task management functions with an I_T nexus scope (e.g., I_T NEXUS RESET) are routed to the task manager in each logical unit about which the task router knows; and
e) Task management functions with I_T_L nexus scope or I_T_L_Q nexus scope addressed to an incorrect logical unit are handled as described in 7.12.

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

4.5.9 SCSI Initiator Device class

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

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

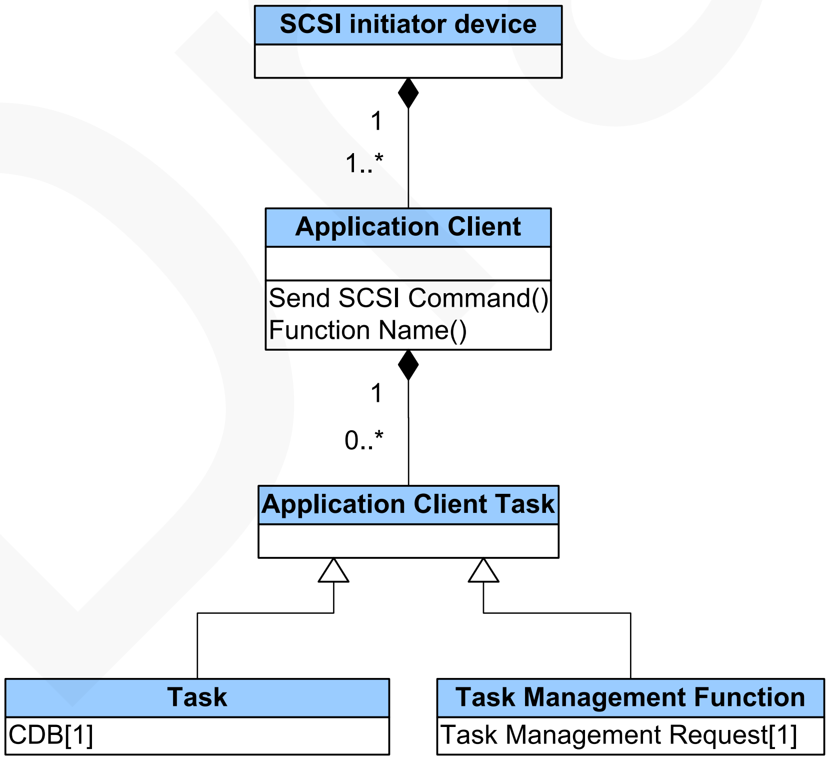


Figure 19 — SCSI Initiator Device class diagram

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 **Send SCSI Command** requests (see 5.4.2).

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

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

4.5.11 Application Client Task class

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

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

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

4.5.12 Task class

4.5.12.1 Task class overview

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

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.

4.5.12.2 CDB Attribute

The CDB attribute contains a CDB (see 5.2 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 7).

4.5.13.2 Task Management Request attribute

The Task Management Request attribute contains a task management request (see clause 7).

4.5.14 SCSI Target Device class

A SCSI Target Device class (see figure 20) is a SCSI Device class that contains the:

a) Level 1 Hierarchical Logical Unit class (see 4.5.15) that contains the:
   A) Logical Unit class (see 4.5.19)
   B) Well Known Logical Unit class (see 4.5.24); or
   C) both.

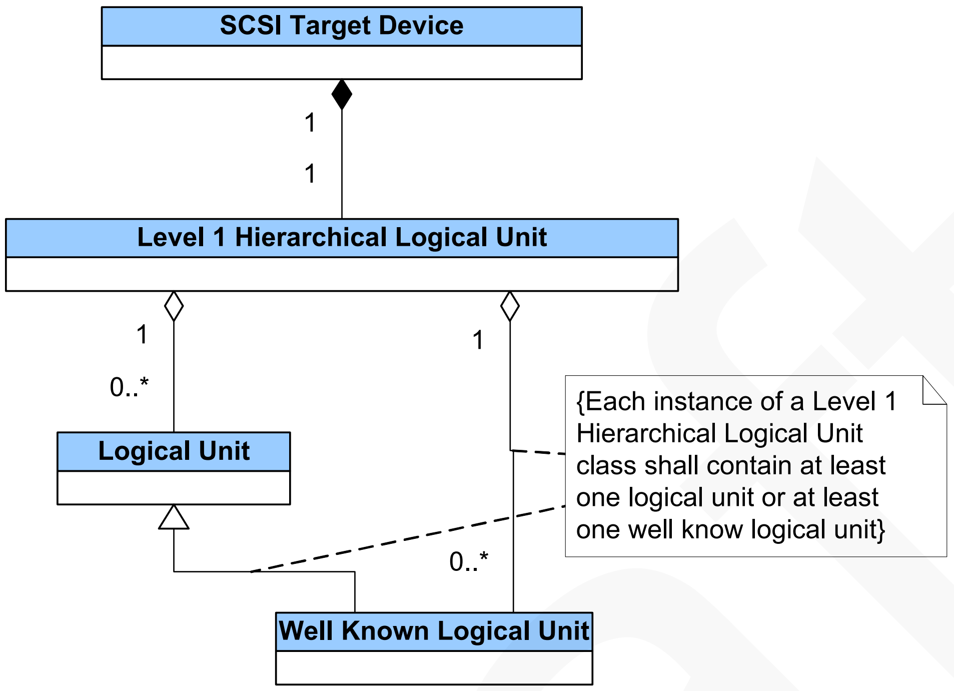

SCSI Target Device
1
1
Level 1 Hierarchical Logical Unit
1
0..*
1
Logical Unit
0..*
Well Known Logical Unit
{Each instance of a Level 1 Hierarchical Logical Unit class shall contain at least one logical unit or at least one well know logical unit}

Figure 20 — 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.

4.5.15 Level 1 Hierarchical Logical Unit class

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

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

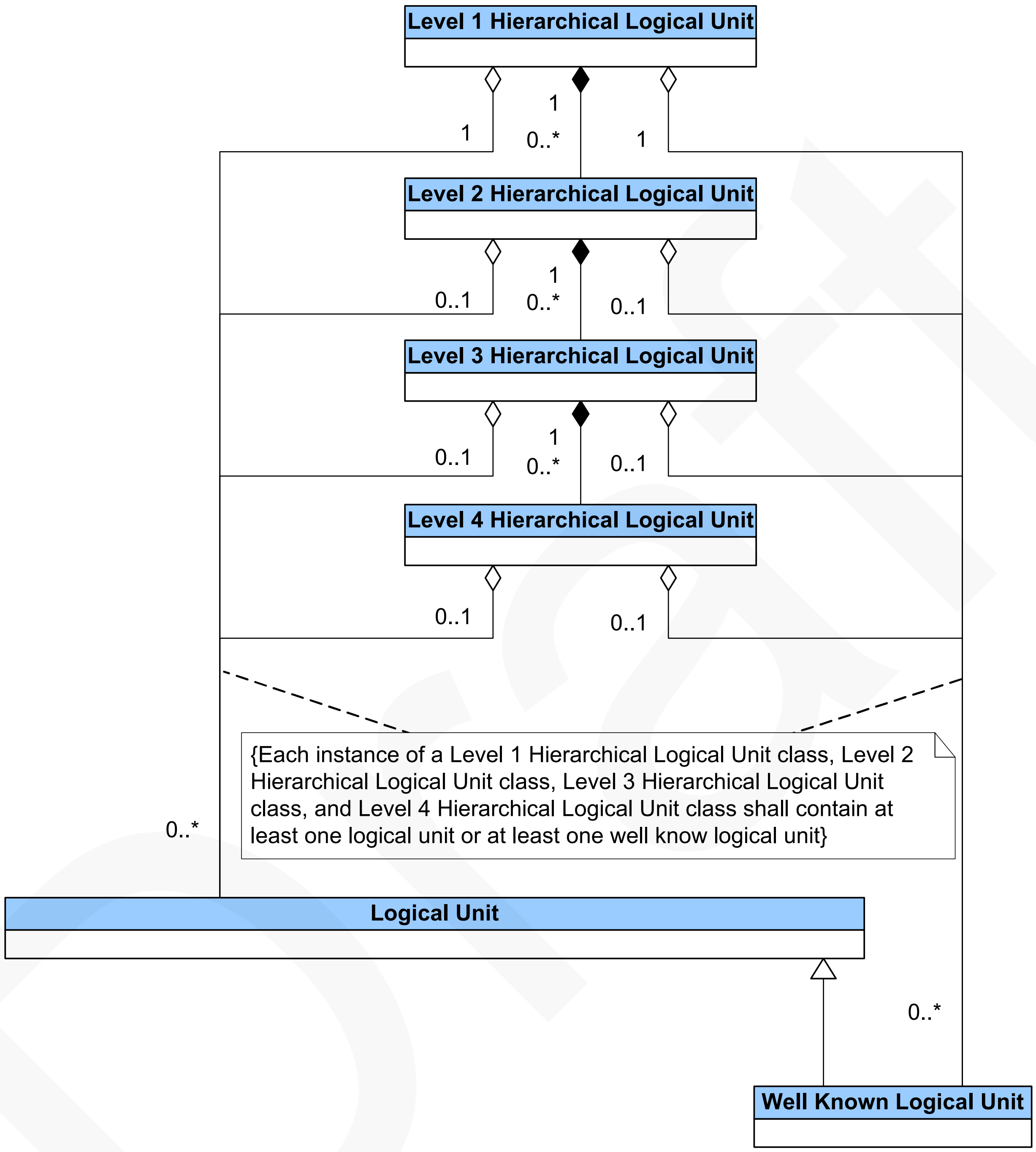

Level 1 Hierarchical Logical Unit
1
1
0..*
1
Level 2 Hierarchical Logical Unit
0..1
1
0..*
0..1
Level 3 Hierarchical Logical Unit
0..1
1
0..*
0..1
Level 4 Hierarchical Logical Unit
0..1
0..1
{Each instance of a Level 1 Hierarchical Logical Unit class, Level 2 Hierarchical Logical Unit class, Level 3 Hierarchical Logical Unit class, and Level 4 Hierarchical Logical Unit class shall contain at least one logical unit or at least one well know logical unit}
0..*
Logical Unit
0..*
Well Known Logical Unit

Figure 21 — 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 4.6.7);
b) Flat space addressing method (see 4.6.8);
c) Logical unit address method (see 4.6.9); or
d) Extended logical unit addressing method (see 4.6.10).

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

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

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

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

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

4.5.16 Level 2 Hierarchical Logical Unit class

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

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

4.5.17 Level 3 Hierarchical Logical Unit class

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

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

4.5.18 Level 4 Hierarchical Logical Unit class

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

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

4.5.19 Logical Unit class

4.5.19.1 Logical Unit class overview

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

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

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

a) Well Known Logical Unit class (see 4.5.19.1); or
b) Hierarchical Logical Unit class.

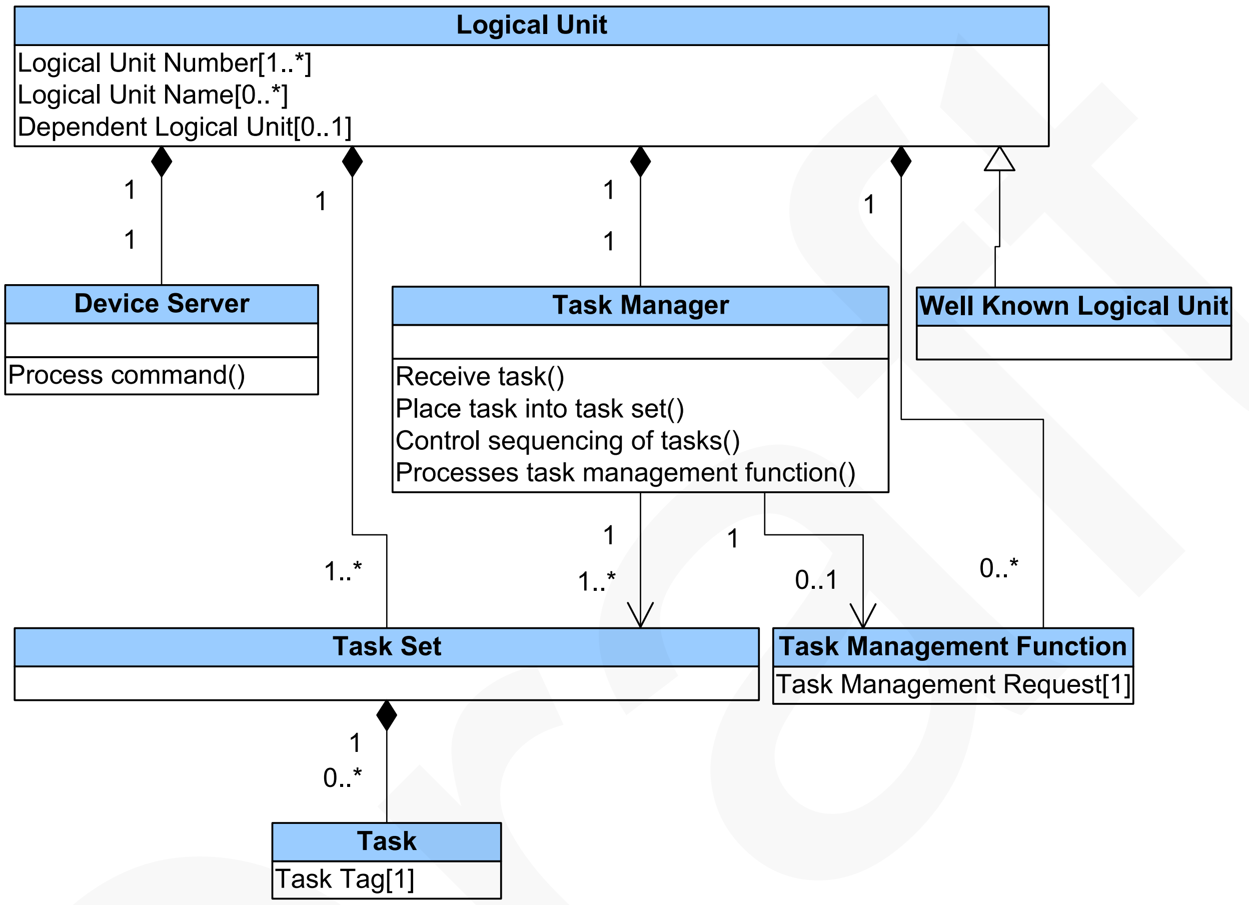


Figure 22 — Logical Unit class diagram

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.

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

4.5.19.2 Logical Unit Number attribute

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

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

Logical unit number(s) are required as follows:

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

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

4.5.19.3 Logical Unit Name attribute

A Logical Unit Name attribute identifies a name (see 3.1.68) for a logical unit that is not a well known logical unit. A logical unit name shall be world wide unique. A logical unit name shall never change and may be used to persistently identify a logical unit.

Logical unit name(s) are required as follows:

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

4.5.19.4 Dependent Logical Unit attribute

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

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

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

4.5.20 Device Server class

The Device Server class processes commands.

4.5.21 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).

4.5.22 Task Set class

A Task Set class contains a Task class (see 4.5.23).

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).

4.5.23 Task class

4.5.23.1 Task class overview

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

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

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

4.5.23.2 Task Tag attribute

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

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

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

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

4.5.24 Well Known Logical Unit class

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

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

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

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

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

All well known logical units:

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

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

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

4.6 Logical unit numbers

4.6.1 Introduction

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

The 64-bit quantity called a LUN is the logical unit number object defined by this standard. The fields containing the acronym LUN that compose the logical unit number object are historical nomenclature anomalies, not logical unit number objects. Logical unit number objects 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).

4.6.2 Logical unit representation format

When an application client displays or otherwise makes a 64-bit LUN value visible to a user, it should display it in hexadecimal format with byte 0 first (i.e., on the left) and byte 7 last (i.e., on the right), regardless of the internal representation of the LUN value (e.g., a single level LUN with an ADDRESS METHOD field set to 01b (i.e., flat space addressing) and a FLAT SPACE LUN field set to 0001h should be displayed as 40 01 00 00 00 00 00 00h, not 00 00 00 00 00 00 01 40h). A separator (e.g., space, dash, or colon) may be included between each byte, each two bytes (e.g., 4001-0000-0000-0000h), or each four bytes (e.g., 40010000 00000000h).

When displaying a single level 64-bit LUN value, an application client may display it as a single 2-byte value representing only the first level LUN (e.g., 40 01h). A separator (e.g., space, dash, or colon) may be included between each byte.

When displaying a 16-bit LUN value, an application client should display it as a single 2-byte value (e.g., 40 01h). A separator (e.g., space, dash, or colon) may be included between each byte.

4.6.3 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 7 in 4.6.6).

4.6.4 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.

4.6.5 Single level logical unit number structure

Table 3 describes a single level subset of the format described in 4.6.6 for SCSI target devices that contain 256 or fewer logical units.

Table 3 — Single level logical unit number structure using peripheral device addressing method

Bit / Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (00b)		BUS IDENTIFIER (00h)					
1	TARGET OR LUN							
2	Null second level LUN (0000h)							
3								
4	Null third level LUN (0000h)							
5								
6	Null fourth level LUN (0000h)							
7								

All logical unit number structure fields beyond byte 1 shall be zero (see table 3). The value in the TARGET OR LUN field shall address a single level logical unit and be between 0 and 255, inclusive. The 00b in the ADDRESS METHOD field specifies peripheral device addressing (see 4.6.6) and the 00h in the BUS IDENTIFIER field specifies the current level (see 4.6.7).

Table 4 describes a single level subset of the format described in 4.6.6 for SCSI target devices that contain 16 384 or fewer logical units.

Table 4 — Single level logical unit number structure using flat space addressing method

Bit / Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (01b)		(MSB) FLAT SPACE LUN					
1	FLAT SPACE LUN							(LSB)
2	Null second level LUN (0000h)							
3								
4	Null third level LUN (0000h)							
5								
6	Null fourth level LUN (0000h)							
7								

All logical unit number structure fields beyond byte 1 shall be zero (see table 4). The value in the FLAT SPACE LUN field shall be between 0 and 16 383, inclusive. The 01b in the ADDRESS METHOD field specifies flat space addressing (see 4.6.8) at the current level.

Table 5 describes a single level subset of the format described in 4.6.6 for SCSI target devices that contain more than 16 384 logical units.

Table 5 — Single level logical unit number structure using extended flat space addressing method

Bit / Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (11b)		LENGTH (01b)		EXTENDED ADDRESS METHOD (2h)			
1	(MSB)							
	EXTENDED FLAT SPACE ADDRESS							
3								(LSB)
8	Null second level LUN (0000h)							
9								
10	Null third level LUN (0000h)							
11								

All logical unit number structure fields beyond byte 3 shall be zero (see table 5). The value in the EXTENDED FLAT SPACE ADDRESS field shall be between 0 and 16 777 215, inclusive. The 11b in the ADDRESS METHOD field with a 2h in the EXTENDED ADDRESS METHOD field specifies extended flat space addressing (see 4.6.12) at the current level. The 01b in the LENGTH field specifies that the LUN specified in the EXTENDED FLAT SPACE ADDRESS field is three bytes in length.

The presence of well-known logical units shall not affect the requirements defined within this subclause.

If a SCSI target device contains 256 or fewer logical units, none of which are dependent logical units (see 4.5.19.4), then the SCSI target device's logical unit numbers:

a) should have the format shown in table 3 (i.e., peripheral device addressing);
b) may have the format shown in table 4 (i.e., flat space addressing); or
c) may have the format shown in table 5 (i.e., extended flat space addressing).

If a SCSI target device contains more than 256 logical units and 16 384 or fewer logical units, none of which are dependent logical units (see 4.5.19.4), then the SCSI target device's logical unit numbers:

a) should have the format shown in table 4 (i.e., flat space addressing);
b) may have the format shown in table 5 (i.e., extended flat space addressing); or
c) may have the format shown in table 3 (i.e., peripheral device addressing) for up to 256 of the logical units within SCSI target device.

If a SCSI target device contains more than 16 384 logical units, none of which are dependent logical units (see 4.5.19.4), then the SCSI target device's logical unit numbers:

a) should have the format shown in table 5 (i.e., extended flat space addressing);
b) may have the format shown in table 4 (i.e., flat space addressing) for up to 16 384 of the logical units within SCSI target device; or
c) may have the format shown in table 3 (i.e., peripheral device addressing) for up to 256 of the logical units within SCSI target device.

4.6.6 Eight byte logical unit number structure

The eight byte logical unit number structure (see table 7) 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 6 and figure 23).

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

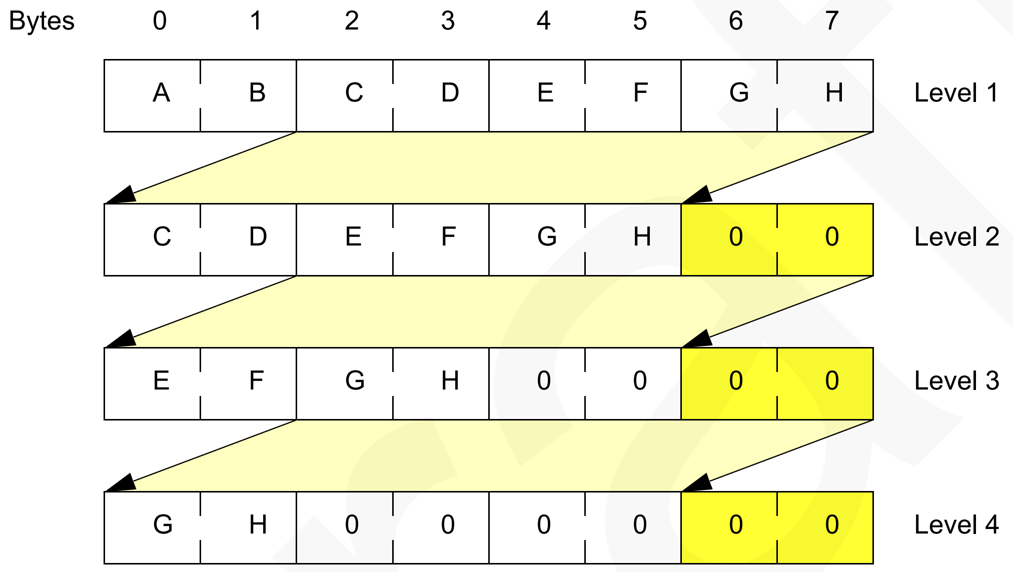


Figure 23 — Eight byte logical unit number structure adjustments

Table 6 — 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 7.

Table 7 — 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 8 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 8 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 8 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 8 for a definition of the FOURTH LEVEL ADDRESSING field.

Table 8 — Format of addressing fields

Bit / Byte	7	6	5	4	3	2	1	0
n-1	ADDRESS METHOD		ADDRESS METHOD SPECIFIC					
n								

The ADDRESS METHOD field defines the contents of the ADDRESS METHOD SPECIFIC field. See table 9 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 9 — ADDRESS METHOD field values

Code	Description	Reference
00b	Peripheral device addressing method	4.6.7
01b	Flat space addressing method	4.6.8
10b	Logical unit addressing method	4.6.9
11b	Extended logical unit addressing method	4.6.10

4.6.7 Peripheral device addressing method

If the peripheral device 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 is not able to 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 10 — 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 4.6.9) 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 4.6.4).

4.6.8 Flat space addressing method

The flat space addressing method (see table 11) 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 11 — Flat space addressing

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

The FLAT SPACE LUN field specifies the current level logical unit.

4.6.9 Logical unit addressing method

If the logical unit addressing method (see table 12) 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 is not able to 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 12 — 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

TARGET OR LUN field in a single level logical unit number structure for logical unit numbers 255 and below (see 4.6.5). The TARGET field contains a mapped representation of a target port identifier.

4.6.10 Extended logical unit addressing

Extended logical unit addressing (see table 13) specifies a logical unit at the current level.

Extended logical unit addressing builds on the formats defined for dependent logical units (see 4.5.19.4) 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 9 in 4.6.6) 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 13 — 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 14) specifies the length of the EXTENDED ADDRESS METHOD SPECIFIC field.

Table 14 — 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 15
01b	3	4	table 16
10b	5	6	table 17
11b	7	8	table 18

Table 15, table 16, table 17, and table 18 show the four extended logical unit addressing formats.

Table 15 — 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 16 — 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 17 — 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 18 — 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 19) specifies the type and size of extended logical unit address found in the EXTENDED ADDRESS METHOD SPECIFIC field.

Table 19 — Logical unit extended addressing

EXTENDED ADDRESS METHOD **Code(s)**	LENGTH **Code(s)**	**Description**	**Reference**
0h	00b - 11b	Reserved	
1h	00b	Well known logical unit	4.6.11
1h	01b - 11b	Reserved	
2h	01b	Extended flat space addressing	4.6.12
2h	00b, 10b, 11b	Reserved	
3h - Eh	00b - 11b	Reserved	
Fh	00b - 10b	Reserved	
Fh	11b	Logical unit not specified	4.6.13

4.6.11 Well known logical unit addressing

A SCSI target device may support zero or more well known logical units (see 4.5.24). 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 20).

Table 20 — 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).

4.6.12 Extended flat space addressing method

The extended flat space addressing method (see table 11) 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 21 — Extended flat space addressing

Bit / Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (01b)		EXTENDED ADDRESS METHOD (2h)			
n+1	(MSB)	EXTENDED FLAT SPACE LUN						
n+3								(LSB)

The EXTENDED FLAT SPACE LUN field specifies a current level logical unit.

4.6.13 Logical unit not specified addressing

Logical unit not specified addressing (see table 22) shall be used to indicate that no logical unit of any kind is specified.

Table 22 — 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 23 maps the nexus object to other identifier objects.

Table 23 — 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 tag	4.5.9 4.5.14 4.5.19 4.5.23.2

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 24.

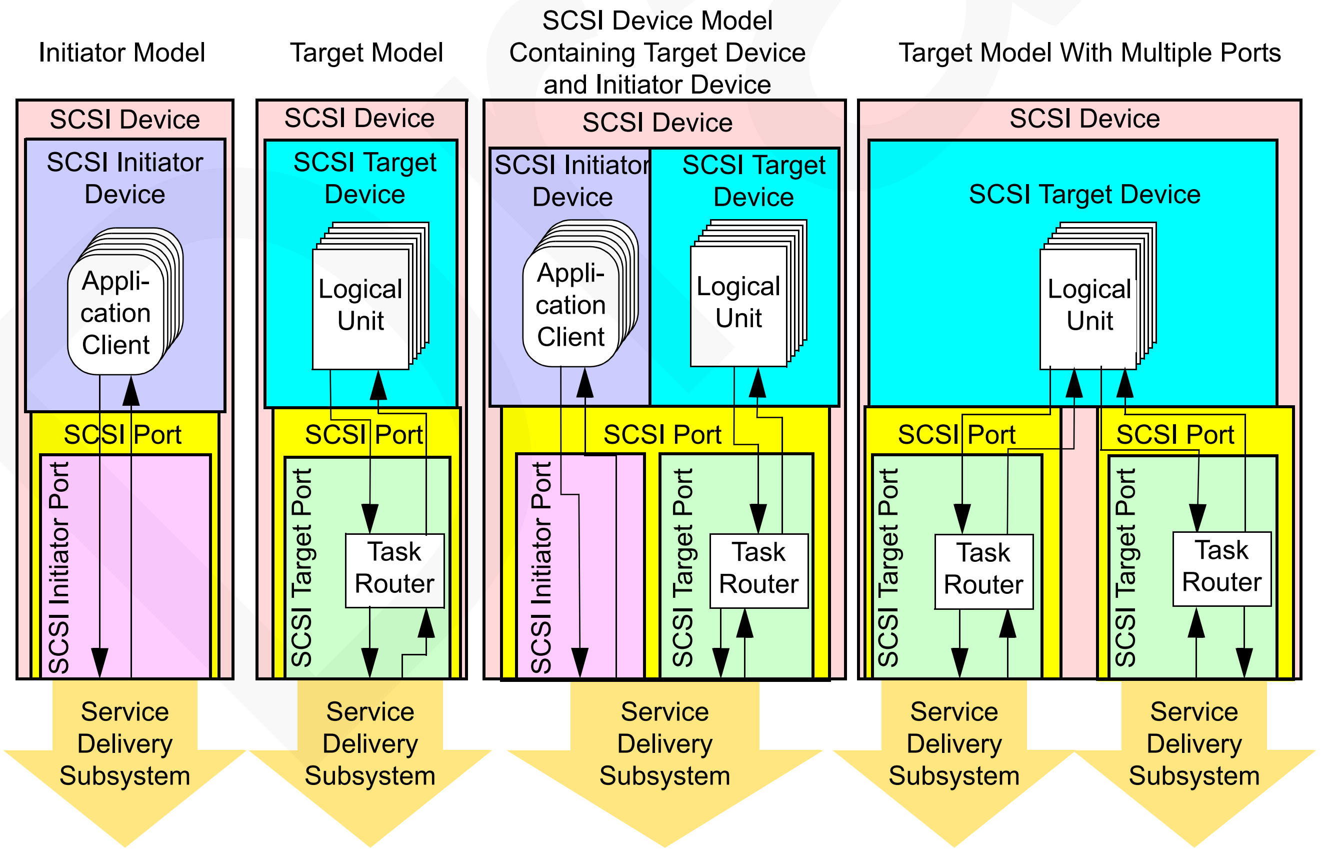


Figure 24 — SCSI device functional models

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 4.5.14) 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 identifiers representing the SCSI ports shall meet the requirements for initiator port identifiers (see 4.5.9) or target port identifiers (see 4.5.14). 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 25 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.

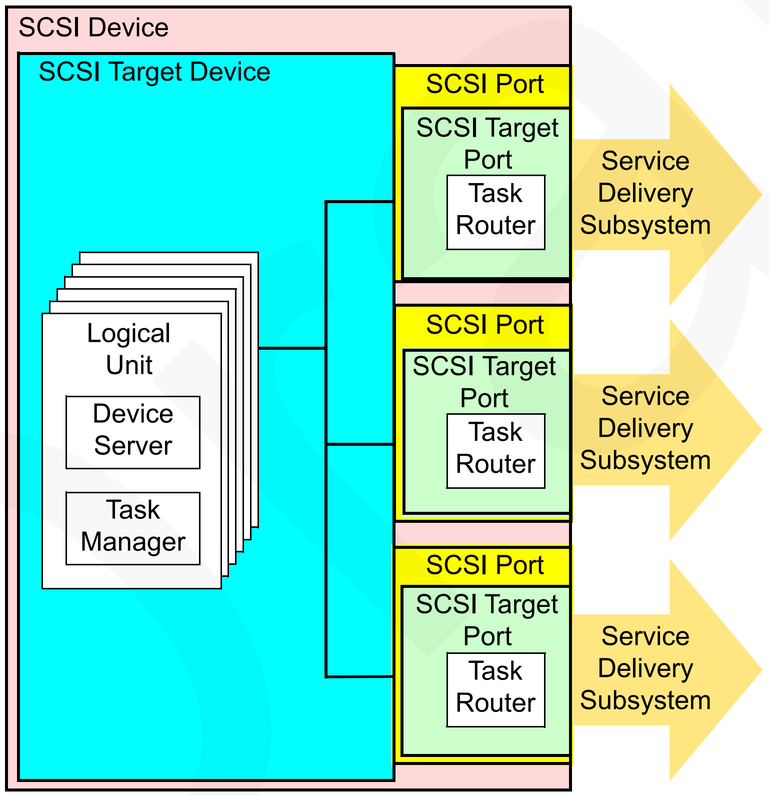


Figure 25 — Multiple port target SCSI device structure model

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 26 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.

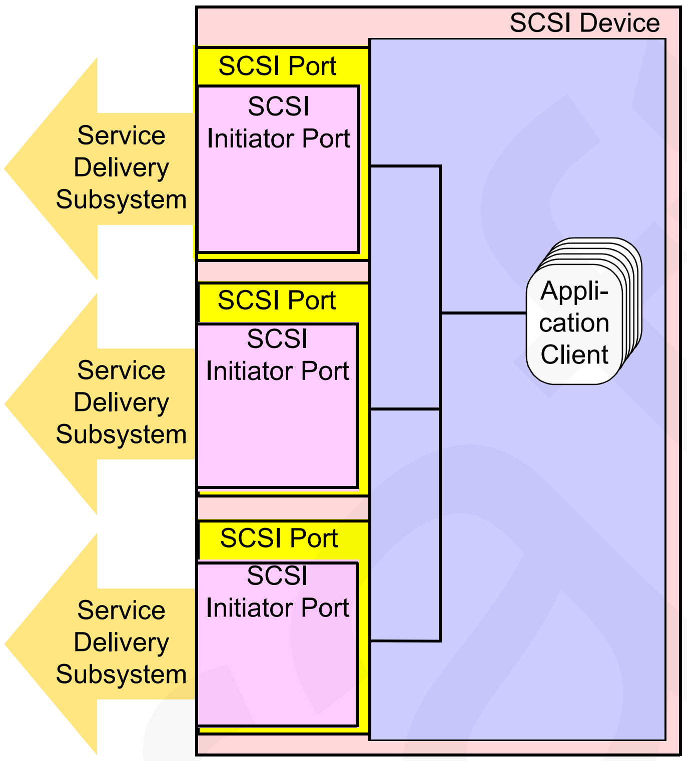


Figure 26 — Multiple port SCSI initiator device structure model

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 27 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.

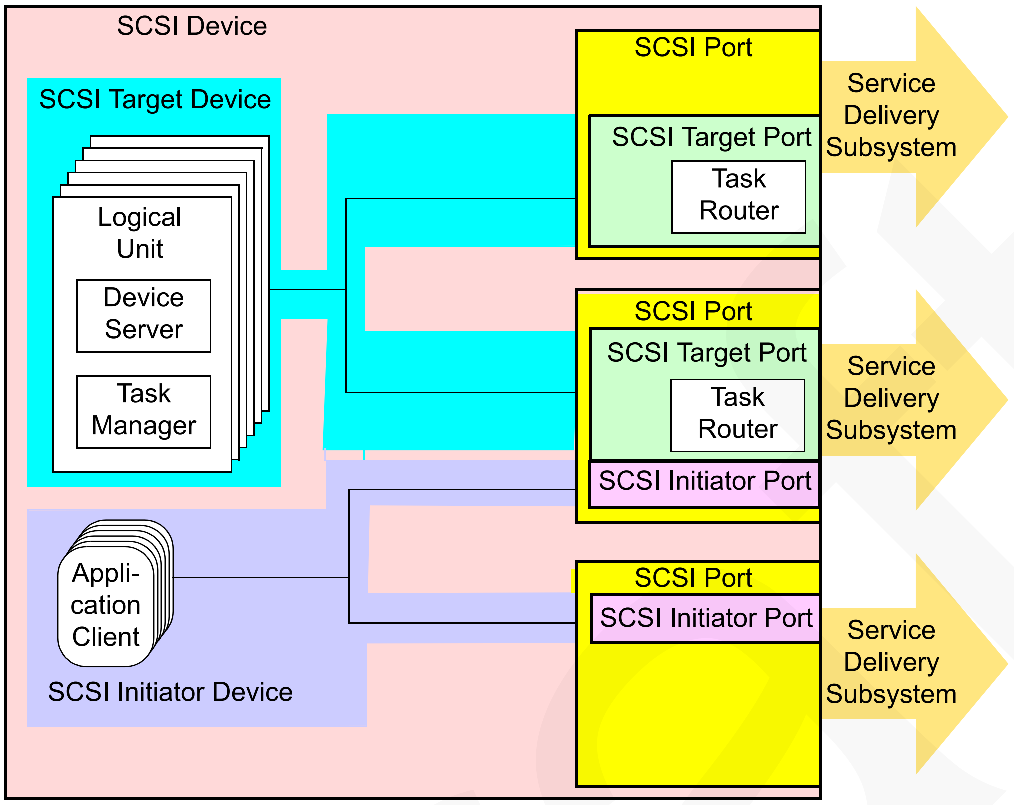


Figure 27 — Multiple port SCSI device structure model

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 28 and figure 29 show two examples of such configurations.

Figure 28 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.

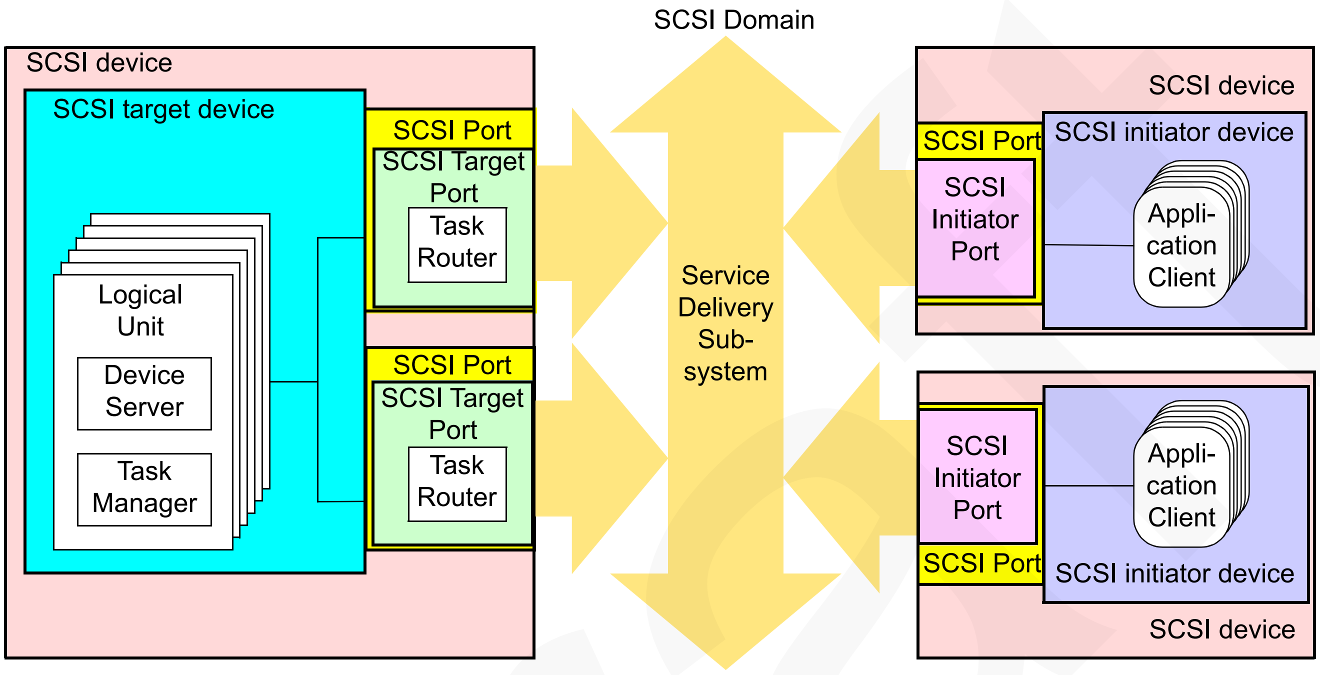


Figure 28 — SCSI target device configured in a single SCSI domain

Figure 29 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 29 and the configuration shown in figure 30.

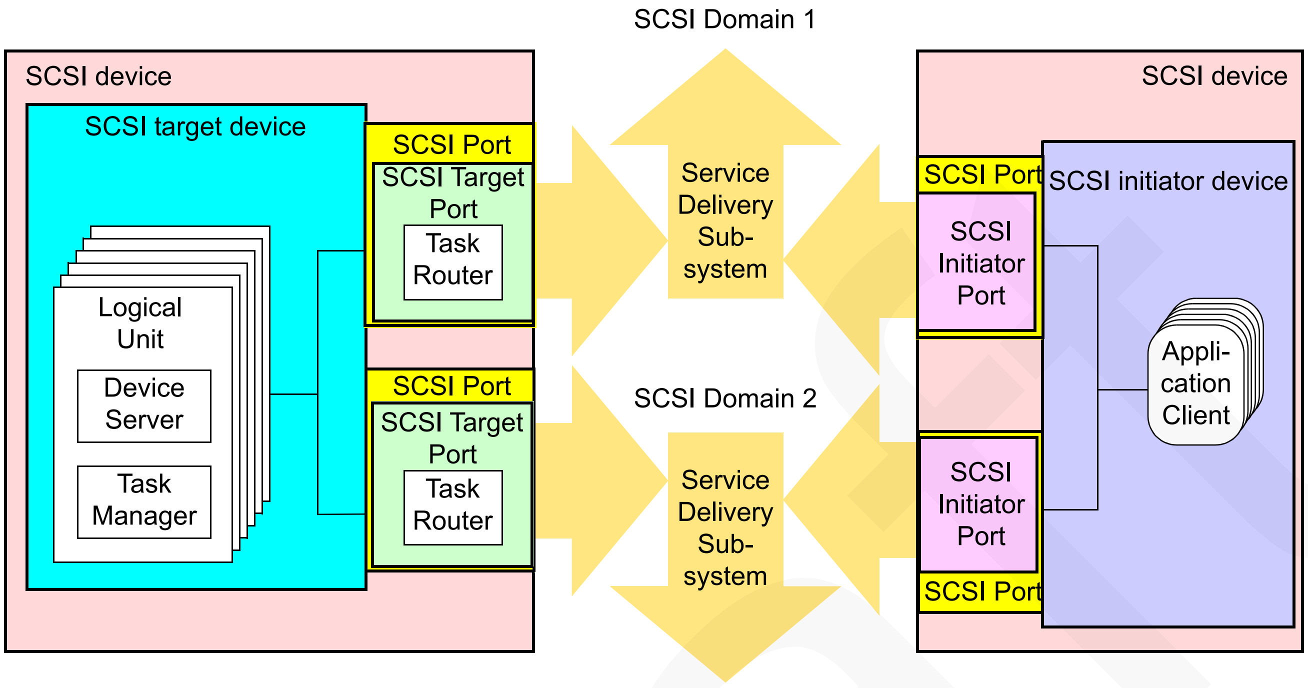


Figure 29 — SCSI target device configured in multiple SCSI domains

Figure 30 shows the same configuration as figure 29 except that the two SCSI domains have been replaced by a single SCSI domain.

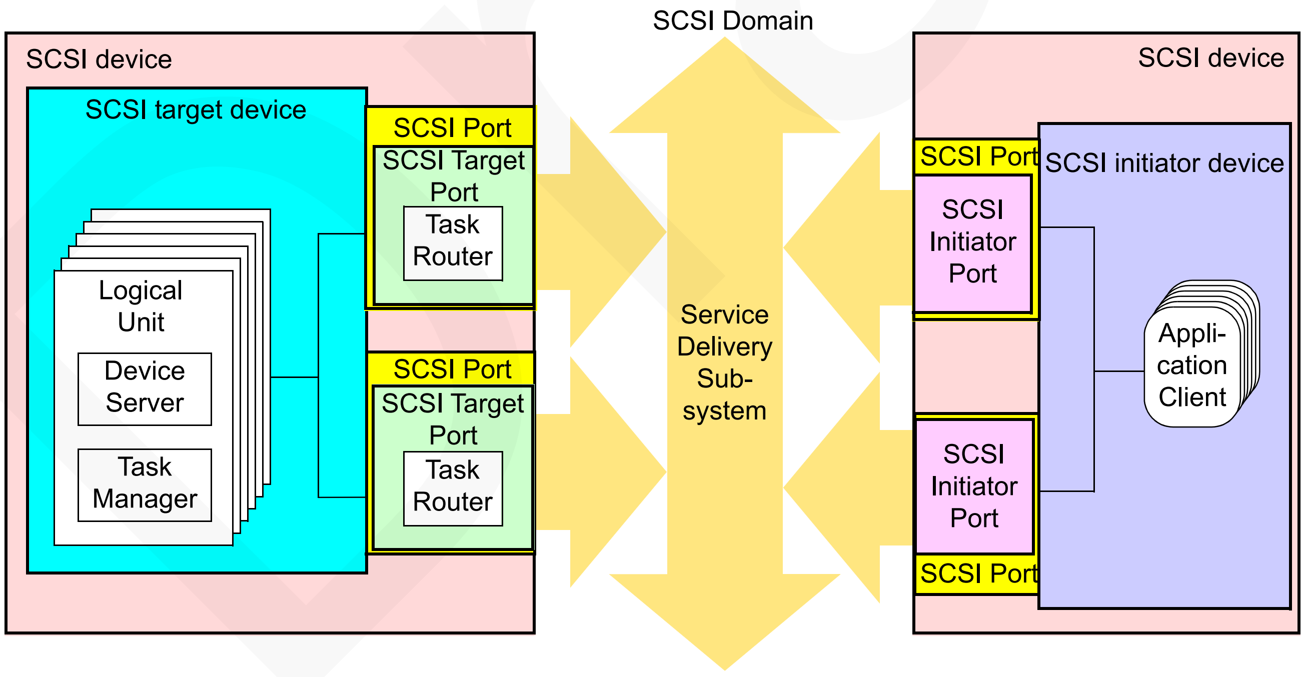


Figure 30 — SCSI target device and SCSI initiator device configured in a single SCSI domain

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 29 and figure 30 in exactly the same way it handles the configuration shown in figure 28).

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 31.

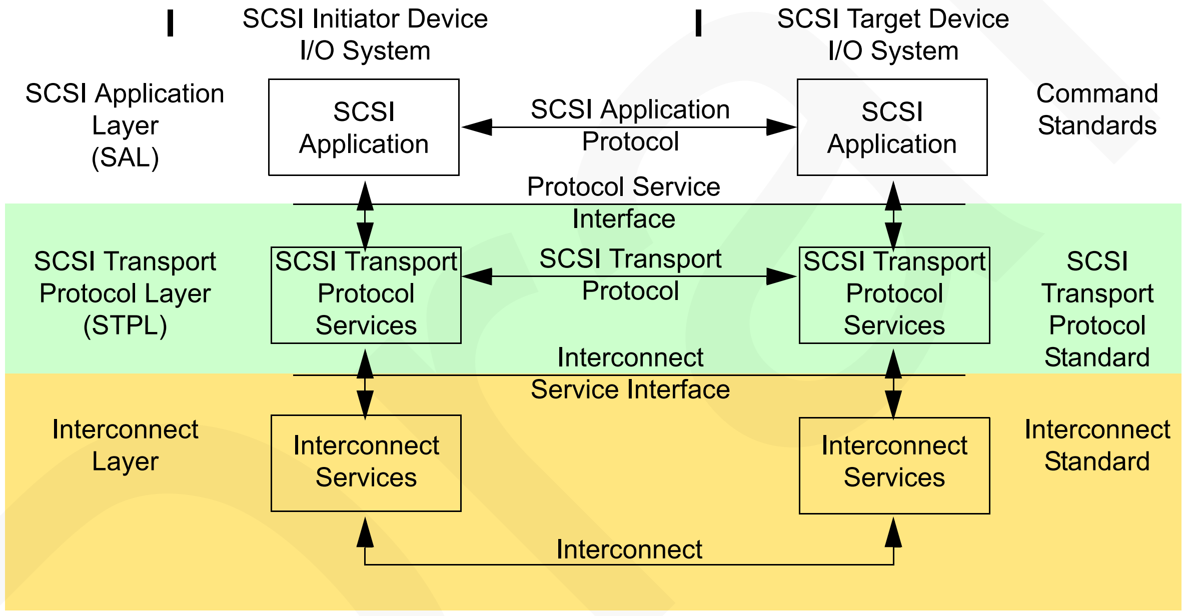

SCSI Initiator Device I/O System
SCSI Target Device I/O System
SCSI Application Layer (SAL)
SCSI Application
SCSI Application Protocol
SCSI Application
Command Standards
Protocol Service Interface
SCSI Transport Protocol Layer (STPL)
SCSI Transport Protocol Services
SCSI Transport Protocol
SCSI Transport Protocol Services
SCSI Transport Protocol Standard
Interconnect Service Interface
Interconnect Layer
Interconnect Services
Interconnect Services
Interconnect Standard
Interconnect

Figure 31 — Protocol service reference model

The layers in 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): Clients and servers that originate and process SCSI I/O operations by means of a SCSI application protocol.
SCSI transport protocol layer (STPL): Services and protocols through which clients and servers communicate.
Interconnect layer: 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 a service delivery subsystem.

The set of SCSI transport protocol services implemented by a 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 3.6.4. 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 32 shows the relationships between the four SCSI transport protocol service types.

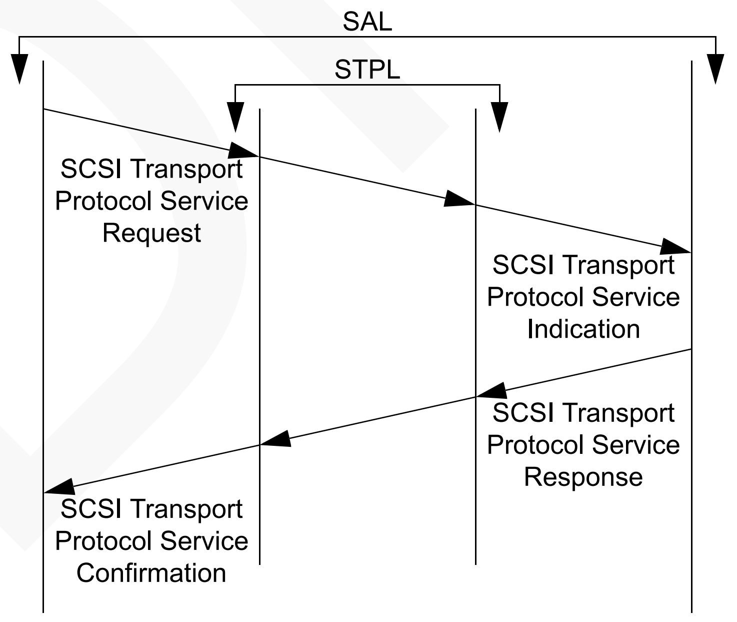

SAL
STPL
SCSI Transport Protocol Service Request
SCSI Transport Protocol Service Indication
SCSI Transport Protocol Service Response
SCSI Transport Protocol Service Confirmation

Figure 32 — SCSI transport protocol service mode

Figure 33 shows how SCSI transport protocol services may be used to process a client-server request-response transaction at the SCSI application layer.

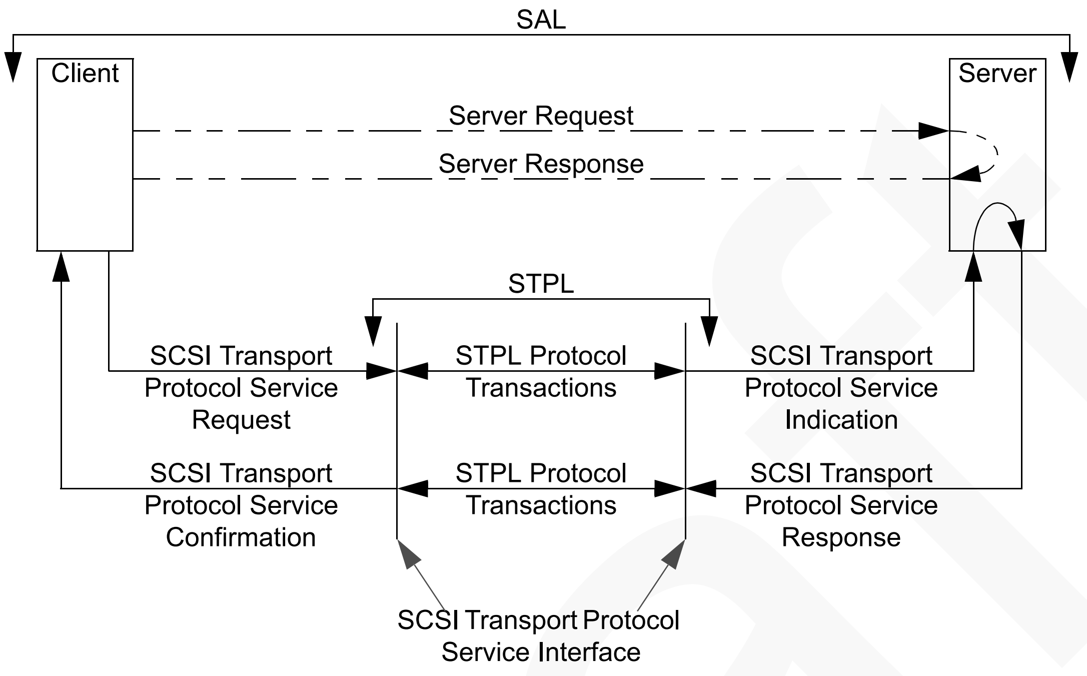

SAL
Client
Server
Server Request
Server Response
STPL
SCSI Transport Protocol Service Request
STPL Protocol Transactions
SCSI Transport Protocol Service Indication
SCSI Transport Protocol Service Confirmation
STPL Protocol Transactions
SCSI Transport Protocol Service Response
SCSI Transport Protocol Service Interface

Figure 33 — Request-Response SAL transaction and related STPL services

The dashed lines in figure 33 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 33 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 34 shows the relationships between the SCSI transport protocol service types involved in a data transfer request.

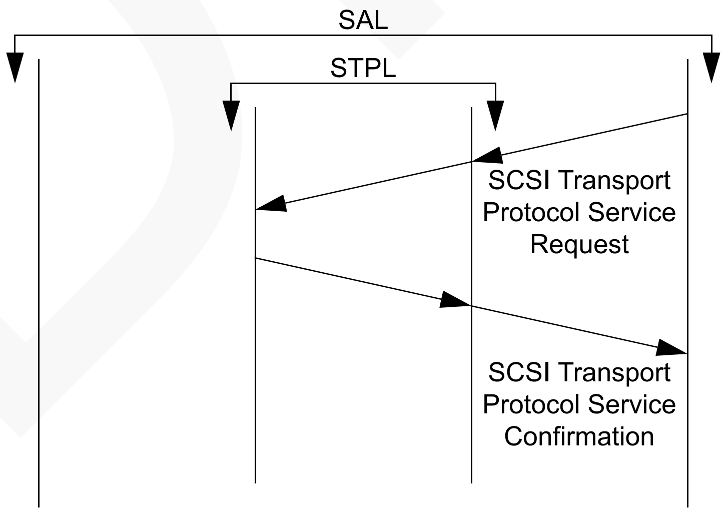

SAL
STPL
SCSI Transport Protocol Service Request
SCSI Transport Protocol Service Confirmation

Figure 34 — SCSI transport protocol service model for data transfers

Figure 35 shows how SCSI transport protocol services may be used to process a device server data transfer transaction.

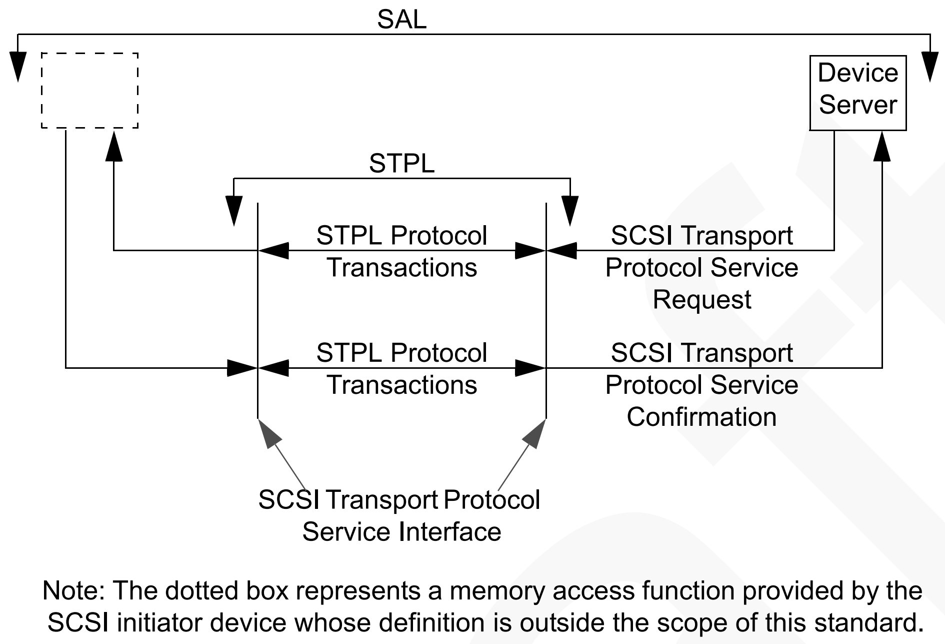

SAL
Device Server
STPL
STPL Protocol Transactions
SCSI Transport Protocol Service Request
STPL Protocol Transactions
SCSI Transport Protocol Service Confirmation
SCSI Transport Protocol Service Interface
Note: The dotted box represents a memory access function provided by the SCSI initiator device whose definition is outside the scope of this standard.

Figure 35 — 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 36 shows the relationships between the SCSI transport protocol service types involved in a Terminate Data Transfer request.

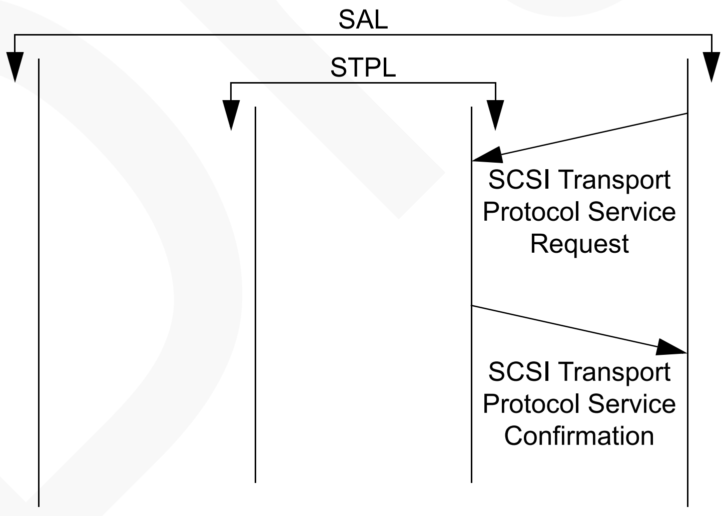

SAL
STPL
SCSI Transport Protocol Service Request
SCSI Transport Protocol Service Confirmation

Figure 36 — SCSI transport protocol service model for Terminate Data Transfer

Figure 37 shows how SCSI transport protocol services may be used to process a device server Terminate Data Transfer transaction.

.

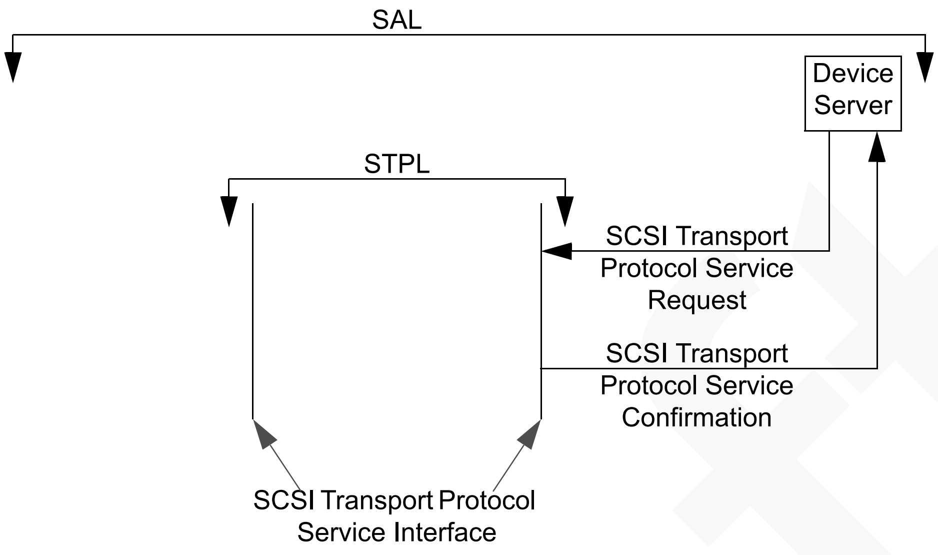

SAL
Device Server
STPL
SCSI Transport Protocol Service Request
SCSI Transport Protocol Service Confirmation
SCSI Transport Protocol Service Interface

Figure 37 — Device server Terminate Data Transfer transaction and related STPL services

5 SCSI command model

5.1 The Execute Command procedure call

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

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

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

Output arguments:

Data-In Buffer:	A buffer to contain command specific information returned by the logical unit by the time of command completion. The **Execute Command** procedure call shall not return a status of GOOD or CONDITION MET unless the buffer contents are valid. The application client shall treat the buffer contents as invalid unless the command completes with a status of GOOD or CONDITION MET. While some valid data may be present for other values of status, the application client should rely on additional information from the logical unit (e.g., sense data) to determine the state of the buffer contents. If the command ends with a service response of SERVICE DELIVERY OR TARGET FAILURE, the application client shall consider the buffer to be undefined.
Sense Data:	A buffer containing sense data returned in the same I_T_L_Q nexus transaction (see 3.1.51) as a CHECK CONDITION status (see 5.8.6). The buffer length is indicated by the Sense Data Length argument. If the command ends with a service response of SERVICE DELIVERY OR TARGET FAILURE, the application client shall consider the sense data to be undefined.
Sense Data Length:	The length in bytes of the Sense Data.
Status:	A one-byte field containing command completion status (see 5.3). If the command ends with a service response of SERVICE DELIVERY OR TARGET FAILURE, the application client shall consider status to be undefined.
Retry Delay Timer:	Additional information about the indicated status code (see 5.3.2).

Service Response assumes one of the following values:

TASK COMPLETE:	A logical unit response indicating that the task has ended. The Status argument shall have one of the values specified in 5.3.
SERVICE DELIVERY OR TARGET FAILURE:	The command has been ended due to a service delivery failure (see 3.1.112) or SCSI target device malfunction. All output parameters are invalid.

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

5.2 Command descriptor block (CDB)

The CDB defines the operation to be performed by the device server.

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

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

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

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

Table 24 — CONTROL byte

Bit	7	6	5	4	3	2	1	0
	Vendor specific		Reserved			NACA	Obsolete	Obsolete

All SCSI transport protocol standards shall define as mandatory the functionality needed for a logical unit to implement the NACA bit.

The NACA (Normal ACA) bit specifies whether an auto contingent allegiance (ACA) is established if the command returns with CHECK CONDITION status. An NACA bit set to one specifies that an ACA shall be established. An NACA bit set to zero specifies that an ACA shall not be established. The actions for ACA are specified in 5.8.2. Actions that may be required when an ACA is not established are described in 5.8.1. All logical units shall implement support for the NACA value of zero and may support the NACA value of one (i.e., ACA). The ability to support a NACA value of one is indicated with the NORMACA bit in the standard INQUIRY data (see SPC-3).

If the NACA bit is set to one but the logical unit does not support ACA, the command shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and the additional sense code set to INVALID FIELD IN CDB.

5.3 Status

5.3.1 Status codes

The status codes are specified in table 25. Status shall be sent from the device server to the application client whenever a command ends with a service response of TASK COMPLETE.

Table 25 — Status codes

Status Code	Status	Task Ended	Service Response
00h	GOOD	Yes	TASK COMPLETE
02h	CHECK CONDITION	Yes	TASK COMPLETE
04h	CONDITION MET	Yes	TASK COMPLETE
08h	BUSY	Yes	TASK COMPLETE
10h	Obsolete		
14h	Obsolete		
18h	RESERVATION CONFLICT	Yes	TASK COMPLETE
22h	Obsolete		
28h	TASK SET FULL	Yes	TASK COMPLETE
30h	ACA ACTIVE	Yes	TASK COMPLETE
40h	TASK ABORTED	Yes	TASK COMPLETE
All other codes	Reserved		

Definitions for each status code are as follows:

GOOD. This status indicates that the device server has successfully completed the task.

CHECK CONDITION. This status indicates that sense data has been delivered in the buffer defined by the Sense Data argument to the **Execute Command** procedure call (see 5.8.6). Additional actions that are required when CHECK CONDITION status is returned are described in 5.8.1.

CONDITION MET. The use of this status is limited to commands for which it is specified (see the PRE-FETCH commands in the SBC-2 standard).

BUSY. This status indicates that the logical unit is busy. This status shall be returned whenever a logical unit is temporarily unable to accept a command. The recommended application client recovery action is to issue the command again at a later time.

If the UA_INTLCK_CTRL field in the Control mode page contains 11b (see SPC-3), termination of a command with BUSY status shall cause a unit attention condition to be established for the SCSI initiator port that sent the command with an additional sense code set to PREVIOUS BUSY STATUS unless a PREVIOUS BUSY STATUS unit attention condition already exists.

Retry delay timer, when supported by a protocol, may provide the SCSI initiator port with more information on when the command should be retransmitted (see table 26).

RESERVATION CONFLICT. This status shall be returned whenever a command attempts to access a logical unit in a way that conflicts with an existing reservation. (See the PERSISTENT RESERVE OUT command and PERSISTENT RESERVE IN command in SPC-3.)

If the UA_INTLCK_CTRL field in the Control mode page contains 11b (see SPC-3), termination of a command with RESERVATION CONFLICT status shall cause a unit attention condition to be established for the SCSI initiator port that sent the command with an additional sense code set to PREVIOUS RESERVATION CONFLICT STATUS unless a PREVIOUS RESERVATION CONFLICT STATUS unit attention condition already exists.

TASK SET FULL. When the logical unit has at least one task in the task set for an I_T nexus and a lack of task set resources prevents accepting a received task from that I_T nexus into the task set, TASK SET FULL status shall be returned. When the logical unit has no task in the task set for an I_T nexus and a lack of task set resources prevents accepting a received task from that I_T nexus into the task set, BUSY status should be returned.

The logical unit should allow at least one command in the task set for each supported I_T nexus (i.e., for each SCSI target port, allow at least one command from each SCSI initiator port that has identified itself to the SCSI target port in a SCSI transport protocol specific manner (e.g., login), or by the successful transmission of a c command).

Retry delay timer, when supported by a protocol, may provide the SCSI initiator port with more information on when the command should be retransmitted (see table 26).

If the UA_INTLCK_CTRL field in the Control mode page contains 11b (see SPC-3), termination of a command with TASK SET FULL status shall cause a unit attention condition to be established for the SCSI initiator port that sent the command with an additional sense code set to PREVIOUS TASK SET FULL STATUS unless a PREVIOUS TASK SET FULL STATUS unit attention condition already exists.

ACA ACTIVE. This status shall be returned as described in 5.8.2.2 and 5.8.2.3 when an ACA exists within a task set. The application client may reissue the command on the same I_T nexus after the ACA condition has been cleared.

TASK ABORTED. This status shall be returned when a task is aborted by another I_T nexus and the Control mode page TAS bit is set to one (see 5.6.3).

5.3.2 Retry delay timer codes

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

Table 26 — Retry delay timer

Status code	Retry delay timer code	Description
BUSY	0000h	No addition information (i.e., the same as normal busy)
	0001h - FFEFh	The number of 100 milliseconds increments the application client should wait before sending another command to the logical unit on any I_T nexus.
	FFF0h - FFFDh	Reserved
	FFFEh	The application client should stop sending commands on this I_T_L nexus.
	FFFFh	The logical unit is not able to accept the command because it is servicing too many other I_T nexus.
TASK SET FULL	0000h	No addition information (i.e., the same as normal task set full)
	0001h - FFEFh	The application client should wait before sending another command to the logical unit on any I_T nexus until: a) at least the number of 100 milliseconds increments indicated in the RETRY DELAY TIMER CODE field have elapsed; or b) a command addressed to the logical unit on any I_T nexus completes.
	FFF0h - FFFFh	Reserved
GOOD	0000h - FFFFh	Reserved
CHECK CONDITION	0000h - FFFFh	Reserved
CONDITION MET	0000h - FFFFh	Reserved
RESERVATION CONFLICT	0000h - FFFFh	Reserved
ACA ACTIVE	0000h - FFFFh	Reserved
TASK ABORTED	0000h - FFFFh	Reserved

5.3.3 Status precedence

If a device server detects that more than one of the following conditions applies to a completed task, it shall select the condition to report based on the following precedence:

1) An ACA ACTIVE status;
2) A CHECK CONDITION status for any of the following unit attention conditions (i.e., with a sense key set to UNIT ATTENTION and one of the following additional sense codes):
   A) POWER ON, RESET, OR BUS DEVICE RESET OCCURRED;
   B) POWER ON OCCURRED;
   C) SCSI BUS RESET OCCURRED;
   D) MICROCODE HAS BEEN CHANGED;
   E) BUS DEVICE RESET FUNCTION OCCURRED;
   F) DEVICE INTERNAL RESET; or
   G) I_T NEXUS LOSS OCCURRED;
3) A RESERVATION CONFLICT status;
   and
4) A status of:
   A) CHECK CONDITION, for any reason not listed in 2);

B) GOOD;
C) CONDITION MET; or
D) TASK ABORTED.

NOTE 5 - The names of the unit attention conditions listed in this subclause (e.g., SCSI BUS RESET OCCURRED) are based on usage in previous versions of this standard. The use of these unit attention condition names is not to be interpreted as a description of how the unit attention conditions are represented by any given SCSI transport protocol.

A device server may report the following status codes with any level of precedence:

a) BUSY status;
b) TASK SET FULL status; or
c) CHECK CONDITION status with a sense key set to ILLEGAL REQUEST.

5.4 SCSI transport protocol services in support of Execute Command

5.4.1 Overview

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

5.4.2 Command and Status SCSI transport protocol services

5.4.2.1 Command and Status SCSI transport protocol services overview

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

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

5.4.2.2 Send SCSI Command transport protocol service request

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

Send SCSI Command transport protocol service request:

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

Input arguments:

I_T_L_Q Nexus:	The I_T_L_Q nexus identifying the task (see 4.7).
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.
Data-In Buffer Size:	The number of bytes available for data transfers to the Data-In Buffer (see 5.4.3). SCSI transport protocols may interpret the Data-In Buffer Size to include both the size and the location of the Data-In Buffer.
Data-Out Buffer:	A buffer containing command specific information to be sent to the logical unit (e.g., data or parameter lists needed to process the command (see 5.1)). The content of the Data-Out Buffer shall not change during the lifetime of the command (see 5.5) as viewed by the application client.
Data-Out Buffer Size:	The number of bytes available for data transfers from the Data-Out Buffer (see 5.4.3).
CRN:	When CRN is used, all sequential commands of an I_T_L nexus shall include a CRN argument that is incremented by one (see 5.1).
Task Priority:	The priority assigned to the task (see 8.7).
First Burst Enabled:	An argument specifying that a SCSI transport protocol specific number of bytes from the Data-Out Buffer shall be delivered to the logical unit without waiting for the device server to invoke the **Receive Data-Out** SCSI transport protocol service.

5.4.2.3 SCSI Command Received transport protocol service indication

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

SCSI Command Received transport protocol service indication:

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

Input arguments:

I_T_L_Q Nexus:	The I_T_L_Q nexus identifying the task (see 4.7).
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.

5.4.2.4 Send Command Complete transport protocol service response

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

Send Command Complete transport protocol service response:

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

Input arguments:

I_T_L_Q Nexus:	The I_T_L_Q nexus identifying the task (see 4.7).
Sense Data:	If present, a Sense Data argument instructs the SCSI target port to return sense data to the SCSI initiator port (see 5.8.6).
Sense Data Length:	The length in bytes of the sense data to be returned to the SCSI initiator port.
Status:	Command completion status (see 5.1).
Service Response:	Possible service response information for the command (see 5.1).
Retry Delay Timer:	The Retry Delay Timer code for the command (see 5.3.2).

5.4.2.5 Command Complete Received transport protocol service confirmation

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

Command Complete Received transport protocol service confirmation:

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

Input arguments:

I_T_L_Q Nexus:	The I_T_L_Q nexus identifying the task (see 4.7).
Data-In Buffer:	A buffer containing command specific information returned by the logical unit on command completion (see 5.1).
Sense Data:	Sense data returned in the same I_T_L_Q nexus transaction (see 3.1.51) as a CHECK CONDITION status (see 5.8.6).
Sense Data Length:	The length in bytes of the received sense data.
Status:	Command completion status (see 5.1).
Service Response:	Service response for the command (see 5.1).
Retry Delay Timer:	The Retry Delay Timer code for the command (see 5.3.2).

5.4.3 Data transfer SCSI transport protocol services

5.4.3.1 Introduction

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

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

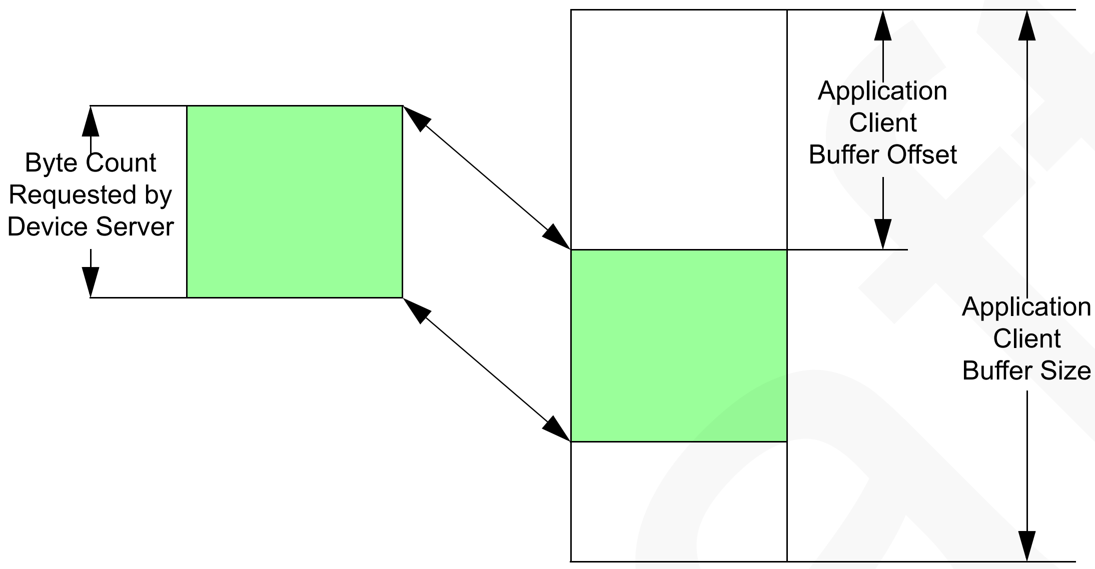


Figure 38 — Model for Data-In and Data-Out data transfers

This standard assumes that the buffering resources available to the logical unit are limited and may be less than the amount of data that is capable of being transferred in one command. Such data needs to be moved between the application client and the media in segments that are smaller than the transfer size specified in the command. The amount of data moved per segment is usually a function of the buffering resources available to the logical unit. Figure 38 shows the model for such incremental data transfers.

SCSI transport protocols may allow logical units to accept the initial portion of the Data-Out Buffer data, called the first burst, along with the command without waiting for the device server to invoke the **Receive Data-Out** SCSI transport protocol service. This is modeled using **Receive Data-Out** protocol service calls for which the SCSI transport protocol may have moved the first burst prior to the call.

SCSI transport protocols that define a first burst capability shall include the First Burst Enabled argument in their definitions for the **Send SCSI Command** and **SCSI Command Received** procedure calls. Logical units that implement the first burst capability shall implement the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see SPC-3).

The movement of data between the application client and device server is controlled by the following arguments:

Application Client Buffer Size:	The total number of bytes in the application client's buffer (i.e., equivalent to Data-In Buffer Size for the Data-In Buffer or equivalent to Data-Out Buffer Size for the Data-Out Buffer).
Application Client Buffer Offset:	Offset in bytes from the beginning of the application client's buffer (Data-In or Data-Out) to the first byte of transferred data.
Byte Count Requested by Device Server:	Number of bytes to be moved by the data transfer request.

For any specific data transfer SCSI transport protocol service request, the **Byte Count Requested by Device Server** is less than or equal to the combination of **Application Client Buffer Size** minus the **Application Client Buffer Offset**.

If a SCSI transport protocol supports random buffer access, the offset and byte count specified for each data segment to be transferred may overlap. In this case the total number of bytes moved for a command is not a reliable indicator of highest byte transferred and shall not be used by a SCSI initiator device or SCSI target device implementation to determine whether all data has been transferred.

All SCSI transport protocol standards shall define support for a resolution of one byte for the Application Client Buffer Size argument.

SCSI transport protocol standards may define restrictions on the resolution of the Application Client Buffer Offset argument. SCSI transport protocol standards may define restrictions on the resolution of the Request Byte Count argument for any call to **Send Data-In** or any call to **Receive Data-Out** that does not transfer the last byte of the Application Client Buffer.

Random buffer access occurs when the device server requests data transfers to or from segments of the application client's buffer that have an arbitrary offset and byte count. Buffer access is sequential when successive transfers access a series of increasing, adjoining buffer segments. Support for random buffer access by a SCSI transport protocol standard is optional. A device server implementation designed for any SCSI transport protocol implementation should be prepared to use sequential buffer access when necessary.

The STPL confirmed services specified in 5.4.3.2 and 5.4.3.3 are used by the device server to request the transfer of data to or from the application client Data-In Buffer or Data-Out Buffer, respectively. The SCSI initiator device SCSI transport protocol service interactions are unspecified.

This standard provides only for the transfer phases to be sequential. Provision for overlapping transfer phases is outside the scope of this standard.

5.4.3.2 Data-In delivery service

5.4.3.2.1 Send Data-In transport protocol service request

A device server uses the Send Data-In transport protocol service request to request that a SCSI target port send data.

Send Data-In transport protocol service request**:**

Send Data-In (IN (I_T_L_Q Nexus, Device Server Buffer, Application Client Buffer Offset, Request Byte Count))

Input argument:

I_T_L_Q Nexus:	The I_T_L_Q nexus identifying the task (see 4.7).
Device Server Buffer:	The buffer in the device server from which data is to be transferred.
Application Client Buffer Offset:	Offset in bytes from the beginning of the application client's buffer (i.e., the Data-In Buffer) to the first byte of transferred data.
Request Byte Count:	Number of bytes to be moved by this request.

5.4.3.2.2 Data-In Delivered transport protocol service confirmation

A SCSI target port uses the Data-In Delivered transport protocol service confirmation to notify a device server that it has sent data.

Data-In Delivered transport protocol service confirmation:

Data-In Delivered (IN (I_T_L_Q Nexus, Delivery Result))

This confirmation notifies the device server that the specified data was successfully delivered to the application client buffer, or that a service delivery subsystem error occurred while attempting to deliver the data.

Input arguments:

I_T_L_Q Nexus:	The I_T_L_Q nexus identifying the task (see 4.7).
Delivery Result:	an encoded value representing one of the following: DELIVERY SUCCESSFUL:The data was delivered successfully. DELIVERY FAILURE:A service delivery subsystem error occurred while attempting to deliver the data.

5.4.3.3 Data-Out delivery service

5.4.3.3.1 Receive Data-Out transport protocol service request

A device server uses the Receive Data-Out transport protocol service request to request that a SCSI target port receive data.

Receive Data-Out transport protocol service request:

Receive Data-Out (IN (I_T_L_Q Nexus, Application Client Buffer Offset, Request Byte Count, Device Server Buffer))

Input arguments:

I_T_L_Q Nexus:	The I_T_L_Q nexus identifying the task (see 4.7).
Device Server Buffer:	The buffer in the device server to which data is to be transferred.
Application Client Buffer Offset:	Offset in bytes from the beginning of the application client's buffer (i.e., the Data-Out Buffer) to the first byte of transferred data.
Request Byte Count:	Number of bytes to be moved by this request.

If the **SCSI Command Received** SCSI transport protocol service included a First Burst Enabled argument and random buffer access is not supported, first burst data shall be transferred to the Device Server Buffer until all first burst data has been transferred. If the **SCSI Command Received** SCSI transport protocol service included a First Burst Enabled argument and random buffer access is supported, first burst data should be transferred to the Device Server Buffer but first burst data may be re-transferred across a service delivery subsystem.

5.4.3.3.2 Data-Out Received transport protocol service confirmation

A SCSI target port uses the Data-Out Received transport protocol service confirmation to notify a device server that it has received data.

Data-Out Received transport protocol service confirmation:

Data-Out Received (IN (I_T_L_Q Nexus, Delivery Result))

This confirmation notifies the device server that the requested data has been successfully delivered to its buffer, or that a service delivery subsystem error occurred while attempting to receive the data.

Input arguments:

I_T_L_Q Nexus:	The I_T_L_Q nexus identifying the task (see 4.7).
Delivery Result:	an encoded value representing one of the following: DELIVERY SUCCESSFUL:The data was delivered successfully. DELIVERY FAILURE:A service delivery subsystem error occurred while attempting to receive the data.

5.4.3.4 Terminate Data Transfer service

5.4.3.4.1 Terminate Data Transfer service overview

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

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

5.4.3.4.2 Terminate Data Transfer transport protocol service request

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

Terminate Data Transfer transport protocol service request:

Terminate Data Transfer (IN (Nexus))

Input argument:

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

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

5.4.3.4.3 Data Transfer Terminated transport protocol service confirmation

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

Data Transfer Terminated transport protocol service confirmation:

Data Transfer Terminated (IN (Nexus))

Input argument:

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

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

5.5 Task and command lifetimes

This subclause specifies the events delimiting the beginning and end (i.e., lifetime) of a task or pending command from the viewpoint of the device server and application client.

The device server shall create a task upon receiving a **SCSI Command Received** indication.

The task shall exist until:

a) The device server sends a SCSI transport protocol service response for the task of TASK COMPLETE; or
b) The task is aborted as described in 5.6.

The application client maintains an application client task to interact with the task from the time the **Send SCSI Command** SCSI transport protocol service request is invoked until it receives one of the following SCSI target device responses:

a) A service response of TASK COMPLETE for that task;
b) Notification of a unit attention condition with one of the following additional sense codes;
    A) Any additional sense code whose ADDITIONAL SENSE CODE field contains 2Fh (e.g., COMMANDS CLEARED BY ANOTHER INITIATOR, or COMMANDS CLEARED BY POWER LOSS NOTIFICATION), if in reference to the task set containing the task;
    B) Any additional sense code whose ADDITIONAL SENSE CODE field contains 29h (e.g., POWER ON, RESET, OR BUS DEVICE RESET OCCURRED; POWER ON OCCURRED; SCSI BUS RESET OCCURRED; BUS DEVICE RESET FUNCTION OCCURRED; DEVICE INTERNAL RESET; or I_T NEXUS LOSS OCCURRED); or
    C) MICROCODE HAS BEEN CHANGED.
c) Notification that the task manager has detected the use of a duplicate I_T_L_Q nexus (see 5.8.3);
d) A service response of FUNCTION COMPLETE following an ABORT TASK task management function directed to the specified task;
e) A service response of FUNCTION COMPLETE following an ABORT TASK SET or a CLEAR TASK SET task management function directed to the task set containing that task;
f) A service response of FUNCTION COMPLETE following an I_T NEXUS RESET task management function delivered on the I_T nexus used to deliver that task; or
g) A service response of FUNCTION COMPLETE in response to a LOGICAL UNIT RESET task management function directed to the logical unit.

If a service response of SERVICE DELIVERY OR TARGET FAILURE is received for a command (e.g., when an I_T nexus loss is detected by the SCSI initiator port), the application client shall maintain an application client task to interact with the task until the application client has determined that the task is no longer known to the device server. An application client may determine that a task is no longer known to the device server by detecting:

a) Completion of an ABORT TASK task management function specifying that task;
b) Completion of an ABORT TASK SET or an I_T NEXUS RESET task management function on the I_T nexus used to deliver that task; or
c) Completion of a CLEAR TASK SET or LOGICAL UNIT RESET task management function.

NOTE 6 - The names of the unit attention conditions listed in the subclause (e.g., SCSI BUS RESET OCCURRED) are based on usage in previous versions of this standard. The use of these unit attention condition names is not to be interpreted as a description of how the unit attention conditions are represented by any given SCSI transport protocol.

To the application client, the command is pending from the time it calls the **Send SCSI Command** SCSI transport protocol service until one of the responses described in this subclause.

When a SCSI transport protocol does not require state synchronization (see 4.3.2), there may be a time skew between the completion of a device server request-response transaction as seen by the application client and device server. As a result, the lifetime of a task or command as it appears to the application client is different from the lifetime observed by the device server.

Some commands (e.g., commands with immediate bits like SEND DIAGNOSTIC, or write commands when a write cache is enabled) start background operations that operate after the task containing the command is no longer in the task set. Background operations may be aborted by power on, hard resets, or logical unit resets. Background operations shall not be aborted by I_T nexus loss.

Background operations may generate deferred errors that are reported in the sense data for a subsequent completed command (see SPC-3). Information that a deferred error occurred may be cleared before it is reported (e.g., by power on, hard reset, or logical unit reset). Deferred errors should not be cleared by I_T nexus loss.

Unless a command completes with a GOOD or CONDITION MET status the degree to which the required command processing has been completed is vendor specific.

5.6 Aborting tasks

5.6.1 Mechanisms that cause tasks to be aborted

A task is aborted when an event or SCSI initiator device action causes termination of the task prior to its successful completion.

The following events cause a task or several tasks to be aborted:

a) The return of an **Execute Command** service response of SERVICE DELIVERY OR TARGET FAILURE as described in 5.1;
b) An I_T nexus loss (see 6.3.4);
c) A logical unit reset (see 6.3.3);
d) A hard reset (see 6.3.2);
e) A power on condition (see 6.3.1);
f) A power loss expected (see 6.3.5); or
g) SCSI transport protocol specific conditions.

An action transmitted via one I_T nexus may abort task(s) received on that I_T nexus and/or task(s) received on other I_T nexuses.

The following actions affect only the task(s) received on the I_T nexus on which the action is transmitted:

a) Completion of an ABORT TASK task management function directed to the specified task;
b) Completion of an ABORT TASK SET task management function under the conditions specified in 7.3;
c) Completion of an I_T NEXUS RESET task management function; or
d) Completion of a command with a CHECK CONDITION status, without establishing an ACA condition (see 5.8.1.3) or establishing an ACA condition (see 5.8.2.2), while the Control mode page (see SPC-3) contains fields that are set as follows:
   A) The QERR field set to 01b and the TST field set to 001b; or
   B) The QERR field set to 11b.

The actions shown in table 27 affect the task(s) received on the I_T nexus on which the action is transmitted and/or task(s) received on other I_T nexuses.

Table 27 — Actions that affect task(s) received on this or other I_T nexuses

Action	Unit attention additional sense code, if any (see 5.6.3)
Completion of a CLEAR TASK SET task management function referencing the task set containing the specified task	COMMANDS CLEARED BY ANOTHER INITIATOR
Completion of a command with a CHECK CONDITION status, with or without establishing an ACA condition, and the QERR field was set to 01b and the TST field was set to 000b in the Control mode page (see SPC-3)	COMMANDS CLEARED BY ANOTHER INITIATOR
Completion of a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action with a reservation key that is associated with the I_T nexus on which the task was received (see SPC-3)	COMMANDS CLEARED BY ANOTHER INITIATOR
Completion of a LOGICAL UNIT RESET task management function (see 7.7) directed to the logical unit	BUS DEVICE RESET FUNCTION OCCURRED
Receipt of a Power Loss Expected indication (see 6.3.5)	COMMANDS CLEARED BY POWER LOSS NOTIFICATION
SCSI transport protocol specific conditions	As defined by the applicable SCSI transport protocol standard

If one or more tasks are cleared or aborted, the affected tasks are also cleared from the SCSI initiator ports in a manner that is outside the scope of this standard.

5.6.2 When a SCSI initiator port aborts tasks received on its own I_T nexus

When a SCSI initiator port causes task(s) received on its own I_T nexus to be aborted, no notification that the task(s) have been aborted shall be returned to the SCSI initiator port other than the completion response for the command or task management function action that caused the task(s) to be aborted and notification(s) associated with related effects of the action (e.g., a reset unit attention condition).

5.6.3 When a SCSI initiator port aborts tasks received on other I_T nexuses

When a SCSI initiator port causes task(s) received on other I_T nexus(es) to be aborted, the SCSI initiator port associated with every other I_T nexus shall be notified that the task(s) have been aborted. The method of notification shall depend on the setting of the TAS bit in the Control mode page (see SPC-3) that applies to the SCSI initiator port(s) associated with the other I_T nexus(es).

If the TAS bit is set to zero, the method of notification shall be a unit attention condition. The additional sense code for the unit attention condition depends on the action that caused the task(s) to be aborted as described in table 27 (see 5.6.1).

If the TAS bit is set to one, the method of notification shall be the termination of each aborted task with a TASK ABORTED status. A unit attention condition containing an additional sense code whose ADDITIONAL SENSE CODE field contains 2Fh (e.g., COMMANDS CLEARED BY ANOTHER INITIATOR or COMMANDS CLEARED BY POWER LOSS NOTIFICATION), shall not be established. The establishment of any other applicable unit attention condition shall not be affected.

When a logical unit is aborting one or more tasks received on an I_T nexus using the TASK ABORTED status it should complete all of those tasks before entering additional tasks received on that I_T nexus into the task set.

5.7 Command processing example

A command is used to show the events associated with the processing of a single device service request (see figure 39). This example does not include error or exception conditions.

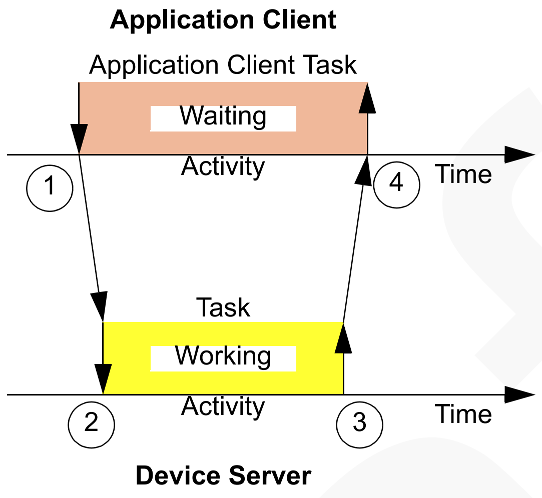


Figure 39 — Command processing events

The numbers in figure 39 identify the events described as follows:

1) The application client task performs an **Execute Command** procedure call by invoking the **Send SCSI Command** SCSI transport protocol service to send the CDB and other input parameters to the logical unit.
2) The device server is notified through a **SCSI Command Received** indication containing the CDB and command parameters. A task is created and entered into the task set. The device server may invoke the appropriate data delivery service one or more times to complete command processing.
3) The task ends upon completion of the command. On command completion, the **Send Command Complete** SCSI transport protocol service is invoked to return a status of GOOD and a service response of TASK COMPLETE.
4) A confirmation of **Command Complete Received** is passed to the application client task by the SCSI initiator port.

5.8 Command processing considerations and exception conditions

5.8.1 Commands that complete with CHECK CONDITION status

5.8.1.1 Overview

When a command completes with a CHECK CONDITION status, the application client may request that the device server alter command processing by establishing an ACA condition, using the NACA bit in the CONTROL byte of the CDB as follows:

a) If the NACA bit is set to zero, an ACA condition shall not be established; or
b) If the NACA bit is set to one, an ACA condition shall be established (see 5.8.2).

The requirements that apply when the ACA condition is not in effect are described in 5.8.1.2.

When a command completes with a CHECK CONDITION status and an ACA condition is not established, tasks other than the task for the command returning the CHECK CONDITION status may be aborted as described in 5.8.1.3.

5.8.1.2 Handling tasks when ACA is not in effect

Table 28 describes the handling of tasks when an ACA condition is not in effect for the task set. Which I_T nexuses are associated with the task set is influenced by the TST field in the Control mode page (see SPC-3).

Table 28 — Task handling when ACA is not in effect

New Task Properties		Device Server Action	ACA Established if New Task Terminates with a CHECK CONDITION status
Attribute [a]	NACA Value [b]		
Any Attribute Except ACA	0	Process the task. [c]	No
	1		Yes
ACA	0	Process an invalid task attribute condition as described in 5.8.5.	No
	1		Yes

a Task attributes are described in 8.6.
b The NACA bit is in the CONTROL byte in the CDB (see 5.2).
c All the conditions that affect the processing of commands (e.g., reservations) apply.

5.8.1.3 Aborting other tasks when CHECK CONDITION status is returned without establishing an ACA

When a CHECK CONDITION status is returned for a command where the NACA bit is set to zero in the command's CDB CONTROL byte (i.e., when an ACA condition is not established), tasks in the dormant or enabled task state (see 8.5) may be aborted based on the contents of the TST field and QERR field in the Control mode page (see SPC-3) as shown in table 29. The TST field specifies the type of task set in the logical unit. The QERR field specifies how the device server handles blocked and dormant tasks when another task receives a CHECK CONDITION status.

Table 29 — Aborting tasks when an ACA is not established

QERR	TST	Action
00b	000b	Tasks other than the task returning CHECK CONDITION status shall not be aborted.
	001b	
01b	000b	All enabled and dormant tasks received on all I_T nexuses shall be aborted (see 5.6).
	001b	All enabled and dormant tasks received on the I_T nexus on which the CHECK CONDITION status was returned shall be aborted (see 5.6). All tasks received on other I_T nexuses shall not be aborted.
11b	000b	All enabled and dormant tasks received on the I_T nexus on which the CHECK CONDITION status was returned shall be aborted (see 5.6). All tasks received on other I_T nexuses shall not be aborted.
	001b	

5.8.2 Auto contingent allegiance (ACA)

5.8.2.1 ACA Overview

When a command completes with a CHECK CONDITION status, the application client may request that the device server alter command processing by establishing an ACA condition, using the NACA bit in the CONTROL byte of the CDB as follows:

a) If the NACA bit is set to zero, an ACA condition shall not be established (see 5.8.1.1); or
b) If the NACA bit is set to one, an ACA condition shall be established.

The steps taken by the device server to establish an ACA condition are described in 5.8.2.2. Upon establishment of the ACA condition, some tasks other than the task returning the CHECK CONDITION status may be aborted and continued processing of other tasks may be blocked as described in 5.8.2.2.

While the ACA condition is in effect and the TMF_ONLY bit is set to zero in the Control mode page (see SPC-3), new tasks received by the logical unit from the faulted I_T nexus are not allowed to enter the task set unless they have the ACA task attribute (see 8.6.5). One of the results of the ACA task attribute requirement is that commands in-flight when the CHECK CONDITION status occurs are returned unprocessed with an ACA ACTIVE status. Multiple commands may be sent one at a time using the ACA task attribute to recover from the event that resulted in the ACA condition without clearing the ACA.

While the ACA condition is in effect and the TMF_ONLY bit is set to one, no new tasks received by the logical unit from the faulted I_T nexus are allowed to enter the task set.

While the ACA condition is in effect:

a) New tasks received on the faulted I_T nexus shall be handled as described in 5.8.2.3, and
b) New tasks received on I_T nexuses other than the faulted I_T nexus shall be handled as described in 5.8.2.4.

The methods for clearing an ACA condition are described in 5.8.2.5.

5.8.2.2 Establishing an ACA

When a device server terminates a command with a CHECK CONDITION status and the NACA bit was set to one in the CONTROL byte of the faulting command, the device server shall create an ACA condition.

When an ACA condition is established, tasks in the dormant or enabled task state (see 8.5) shall either be aborted or blocked based on the contents of the TST field and QERR field in the Control mode page (see SPC-3) as shown in table 30. The TST field specifies the type of task set in the logical unit. The QERR field specifies how the device server handles blocked and dormant tasks when another task receives a CHECK CONDITION status.

Table 30 — Blocking and aborting tasks when an ACA is established

QERR	TST	Action
00b	000b	All enabled tasks received on all I_T nexuses shall transition to the blocked task state (see 8.8). All dormant tasks received on all I_T nexuses shall remain in the dormant task state.
	001b	All enabled tasks received on the faulted I_T nexus shall transition to the blocked task state (see 8.8). All dormant tasks received on the faulted I_T nexus shall remain in the dormant task state. All tasks received on I_T nexuses other than the faulted I_T nexus shall not be affected by the establishment of this ACA condition.
01b	000b	All enabled and dormant tasks received on all I_T nexuses shall be aborted (see 5.6).
	001b	All enabled and dormant tasks received on the faulted I_T nexus shall be aborted (see 5.6). All tasks received on I_T nexuses other than the faulted I_T nexus shall not be affected by the establishment of this ACA condition.
11b	000b	All enabled and dormant tasks received on the faulted I_T nexus shall be aborted (see 5.6). All enabled tasks received on I_T nexuses other than the faulted I_T nexus shall transition to the blocked task state (see 8.8). All dormant tasks received on I_T nexuses other than the faulted I_T nexus shall remain in the dormant task state.
	001b	All enabled and dormant tasks received on the faulted I_T nexus shall be aborted (see 5.6). All tasks received on I_T nexuses other than the faulted I_T nexus shall not be affected by the establishment of this ACA condition.

An ACA condition shall not cross task set boundaries and shall be preserved until it is cleared as described in 5.8.2.5.

If the SCSI transport protocol does not enforce state synchronization as described in 4.5.14, there may be a time delay between the occurrence of the ACA condition and the time at which the application client becomes aware of the condition.

5.8.2.3 Handling new tasks received on the faulted I_T nexus when ACA is in effect

Table 31 describes the handling of new tasks received on the faulted I_T nexus when ACA is in effect.

Table 31 — Handling for new tasks received on a faulted I_T nexus during ACA

New Task Properties		ACA Task Present in the Task Set	TMF_ONLY value [c]	Device Server Action	ACA Established If New Task Terminates with a CHECK CONDITION status
Attribute [a]	NACA Value [b]				
ACA	0	No	0	Process the task. [e]	No [d]
	1	No	0		Yes [d]
	n/a	n/a	1	Terminate the task with ACA ACTIVE status.	n/a
	0 or 1	Yes	n/a		n/a
Any Attribute Except ACA	0 or 1	n/a	n/a	Terminate the task with ACA ACTIVE status.	n/a

a Task attributes are described in 8.6.
b The NACA bit is in the CONTROL byte in the CDB (see 5.2).
c The TMF_ONLY bit is in the Control mode page (see SPC-3).
d If a task with the ACA attribute terminates with a CHECK CONDITION status, the existing ACA condition shall be cleared and the value of the NACA bit shall control the establishment of a new ACA condition.
e All the conditions that affect the processing of commands (e.g., reservations) apply.

5.8.2.4 Handling new tasks received on non-faulted I_T nexuses when ACA is in effect

5.8.2.4.1 Command processing permitted for tasks received on non-faulted I_T nexuses during ACA

The device server shall process a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action (see SPC-3) while an ACA condition is established when the command is received on a non-faulted I_T nexus.

NOTE 7 - The processing of specific commands (e.g., PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action) received on a non-faulted I_T nexus while an ACA condition is in effect provides SCSI initiator ports not associated with the faulted I_T nexus the opportunity to recover from error conditions that the initiator port associated with the faulted I_T nexus is unable to recover from itself.

5.8.2.4.2 Handling new tasks received on non-faulted I_T nexuses when ACA is in effect

The handling of tasks received on I_T nexuses other than the faulted I_T nexus depends on the value in the TST field in the Control mode page (see SPC-3).

Table 32 describes the handling of new tasks received on I_T nexuses other than the faulted I_T nexus when ACA is in effect.

Table 32 — Handling for new tasks received on non-faulted I_T nexuses during ACA

TST Field Value in Control mode page	New Task Properties		New Command Permitted During ACA [c]	Device Server Action	ACA Established If New Task Terminates with a CHECK CONDITION status
	Attribute [a]	NACA Value [b]			
000b	ACA	n/a	n/a	Terminate the task with ACA ACTIVE status.	n/a
	Any Attribute Except ACA	0	No	Terminate the task with BUSY status.	n/a
		1	No	Terminate the task with ACA ACTIVE status.	n/a
		0	Yes	Process the task.	No [d]
		1	Yes		Yes [d]
001b	ACA	0	n/a	Process an invalid task attribute condition as described in 5.8.5.	No
		1			Yes
	Any Attribute Except ACA	0 or 1	n/a	Process the task. [e]	See 5.8.1.2.

a Task attributes are described in 8.6.
b The NACA bit is in the CONTROL byte in the CDB (see 5.2).
c See 5.8.2.4.1.
d If a permitted command terminates with a CHECK CONDITION status, the existing ACA condition shall be cleared and the value of the NACA bit shall control the establishment of a new ACA condition.
e When the TST field in the Control mode page contains 001b, commands received on a non-faulted I_T nexus shall be processed as if the ACA condition does not exist (see 5.8.1.2). In this case, the logical unit shall be capable of handling concurrent ACA conditions and sense data associated with each I_T nexus.

5.8.2.5 Clearing an ACA condition

An ACA condition shall only be cleared:

a) As a result of a hard reset (see 6.3.2), logical unit reset (see 6.3.3), or I_T nexus loss (see 6.3.4);
b) By a CLEAR ACA task management function (see 7.4) received on the faulted I_T nexus;
c) By a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action with the ACA task attribute received on the faulted I_T nexus that clears the tasks received on the faulted I_T nexus (see SPC-3);
d) By a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action with a task attribute other than ACA task attribute received on a non-faulted I_T nexus that clears the tasks received on the faulted I_T nexus;
e) When a command with the ACA task attribute received on the faulted I_T nexus terminates with a CHECK CONDITION status; or

f) When a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action terminates in a CHECK CONDITION status.

Cases e) and f) may result in the establishment of a new ACA based on the value of the NACA bit.

When an ACA condition is cleared and no new ACA condition is established, the state of all tasks in the task set shall be modified as described in 8.8.

5.8.3 Overlapped commands

An overlapped command occurs when a task manager or a task router detects the use of a duplicate I_T_L_Q nexus (see 4.5.6) in a command before a task holding that I_T_L_Q nexus completes its task lifetime (see 5.5). Each SCSI transport protocol standard shall specify whether or not a task manager or a task router is required to detect overlapped commands.

A task manager or a task router that detects an overlapped command shall abort all tasks received on the I_T nexus on which the overlapped command was received and the device server shall return CHECK CONDITION status for the overlapped command. The sense key shall be set to ABORTED COMMAND and the additional sense code shall be set to OVERLAPPED COMMANDS ATTEMPTED.

NOTE 8 - An overlapped command may be indicative of a serious error and, if not detected, may result in corrupted data. This is considered a catastrophic failure on the part of the SCSI initiator device. Therefore, vendor specific error recovery procedures may be required to guarantee the data integrity on the medium. The SCSI target device logical unit may return additional sense data to aid in this error recovery procedure (e.g., sequential-access devices may return the residue of blocks remaining to be written or read at the time the second command was received).

5.8.4 Incorrect logical unit selection

The SCSI target device's response to a command addressed to an incorrect logical unit number is described in this subclause.

In response to a REQUEST SENSE command, a REPORT LUNS command, or an INQUIRY command the SCSI target device shall respond as defined in SPC-3.

Any command except REQUEST SENSE, REPORT LUNS, or INQUIRY:

a) Shall be terminated with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and with the additional sense code set to LOGICAL UNIT NOT SUPPORTED, if:
   A) The SCSI target device is not capable of supporting the logical unit (e.g., some SCSI target devices support only one peripheral device); or
   B) The SCSI target device supports the logical unit, but the peripheral device is not currently connected to the SCSI target device;

   or
b) Is responded to in a vendor specific manner, if:
   A) The SCSI target device supports the logical unit and the peripheral device is connected, but the peripheral device is not operational; or
   B) The SCSI target device supports the logical unit but is incapable of determining if the peripheral device is connected or is not operational because the peripheral device is not ready.

5.8.5 Task attribute exception conditions

If a command is received with a task attribute that is not supported or is not valid (e.g., an ACA task attribute when an ACA condition does not exist), the command shall be terminated with CHECK CONDITION status, sense key set to ILLEGAL REQUEST, and additional sense code set to INVALID MESSAGE ERROR.

NOTE 9 - The use of the INVALID MESSAGE ERROR additional sense code is based on its similar usage in previous versions of this standard. The use of the INVALID MESSAGE ERROR additional sense code is not to be interpreted as a description of how the task attributes are represented by any given SCSI transport protocol.

Task attribute support should be reported with the Extended INQUIRY Data VPD page (see SPC-3).

5.8.6 Sense data

Sense data shall be made available by the logical unit in the event a command completes with a CHECK CONDITION status or other conditions (e.g., the processing of a REQUEST SENSE command). The format, content, and conditions under which sense data shall be prepared by the logical unit are specified in this standard, SPC-3, the applicable command standard, and the applicable SCSI transport protocol standard.

Sense data associated with an I_T nexus shall be preserved by the logical unit until:

a) The sense data is transferred;
b) A logical unit reset (see 6.3.3) occurs;
c) Power loss expected (see 6.3.5) occurs; or
d) An I_T nexus loss (see 6.3.4) occurs for the I_T nexus associated with the preserved sense data.

When a command completes with a CHECK CONDITION status, sense data shall be returned in the same I_T_L_Q nexus transaction (see 3.1.51) as the CHECK CONDITION status. After the sense data is returned, it shall be cleared except when it is associated with a unit attention condition and the UA_INTLCK_CTRL field in the Control mode page (see SPC-3) contains 10b or 11b.

The return of sense data in the same I_T_L_Q nexus transaction as a CHECK CONDITION status shall not affect ACA (see 5.8.2) or the sense data associated with a unit attention condition when the UA_INTLCK_CTRL field contains 10b or 11b.

5.8.7 Unit Attention condition

Each logical unit shall generate a unit attention condition whenever one of the following events occurs:

a) A hard reset (see 6.3.2), logical unit reset (see 6.3.3), I_T nexus loss (see 6.3.4), or power loss expected (see 6.3.5) occurs;
b) A removable medium may have been changed;
c) The mode parameters associated with this I_T nexus have been changed by a task received on another I_T nexus (i.e., SCSI initiator ports share mode parameters, see SPC-3);
d) The log parameters associated with this I_T nexus have been changed by a task received on another I_T nexus (i.e., SCSI initiator ports share log parameters, see SPC-3);
e) The version or level of microcode has been changed (see SPC-3);
f) Tasks received on this I_T nexus have been cleared by a task or a task management function associated with another I_T nexus and the TAS bit was set to zero in the Control mode page associated with this I_T nexus (see SPC-3);
g) INQUIRY data has been changed (see SPC-3);
h) The logical unit inventory has been changed (see 4.5.19.1);
i) The mode parameters in effect for the associated I_T nexus have been restored from non-volatile memory (see SPC-3); or
j) Any other event requiring the attention of the SCSI initiator device.

Logical units may queue unit attention conditions. After the first unit attention condition is cleared, another unit attention condition may exist (e.g., a unit attention condition with an additional sense code set to POWER ON OCCURRED may be followed by one with an additional sense code set to MICROCODE HAS BEEN CHANGED).

A unit attention condition shall persist on the logical unit for the SCSI initiator port associated with each I_T nexus until the SCSI initiator port associated with the I_T nexus clears the condition. Unit attention conditions are affected by the processing of commands as follows:

a) If an INQUIRY command enters the enabled task state, the device server shall perform the INQUIRY command and shall neither report nor clear any unit attention condition;
b) If a REPORT LUNS command enters the enabled task state, the device server shall perform the REPORT LUNS command and shall not report any unit attention condition.

If the UA_INTLCK_CTRL field in the Control mode page is set to 00b (see SPC-3), the SCSI target device shall clear any pending unit attention condition with an additional sense code of REPORTED LUNS DATA HAS CHANGED established for the initiator port associated with that I_T nexus in each logical unit accessible by the I_T nexus on which the REPORT LUNS command was received. Other pending unit attention conditions shall not be cleared.

If the UA_INTLCK_CTRL field in the Control mode page is not set to 00b, the SCSI target device shall not clear any unit attention condition(s);

c) If a REQUEST SENSE command enters the enabled task state while a unit attention condition exists for the SCSI initiator port associated with the I_T nexus on which the REQUEST SENSE command was received, then the device server shall return GOOD status and either:
   A) Report any pending sense data as parameter data and preserve all unit attention conditions on the logical unit; or
   B) Report a unit attention condition as parameter data for the REQUEST SENSE command to the SCSI initiator port associated with the I_T nexus on which the REQUEST SENSE command was received. The logical unit may discard any pending sense data and shall clear the reported unit attention condition for the SCSI initiator port associated with that I_T nexus. If the unit attention condition has an additional sense code of REPORTED LUNS DATA HAS CHANGED, the SCSI target device shall clear any pending unit attention conditions with an additional sense code of REPORTED LUNS DATA HAS CHANGED established for the I_T nexus on which the command was received in each logical unit accessible by that I_T nexus;

   If the device server has already generated the ACA condition (see 5.8.2) for a unit attention condition, the device server shall report the unit attention condition (i.e., option c)B) above); and

d) If a command other than INQUIRY, REPORT LUNS, or REQUEST SENSE enters the enabled task state while a unit attention condition exists for the SCSI initiator port associated with the I_T nexus on which the command was received, the device server shall terminate the command with a CHECK CONDITION status. The device server shall provide sense data that reports a unit attention condition for the SCSI initiator port that sent the command on the I_T nexus.

   If a device server reports a unit attention condition with a CHECK CONDITION status and the UA_INTLCK_CTRL field in the Control mode page contains 00b (see SPC-3), then the device server shall clear the reported unit attention condition for the SCSI initiator port associated with that I_T nexus on the logical unit. If the unit attention condition has an additional sense code of REPORTED LUNS DATA HAS CHANGED, the SCSI target device shall clear any pending unit attention conditions with an additional sense code of REPORTED LUNS DATA HAS CHANGED established for the I_T nexus on which the command was received in each logical unit accessible by that I_T nexus. If the UA_INTLCK_CTRL field contains 10b or 11b, the device server shall not clear unit attention conditions reported with a CHECK CONDITION status.

6 SCSI events and event notification model

6.1 SCSI events overview

SCSI events may occur or be detected in either:

a) The SCSI device;
b) One or more SCSI ports within a SCSI device; or
c) The application client, task manager, or device server.

The detection of any event may require processing by the object that detects it.

Events that occur in the SCSI device are assumed to be detected and processed by all objects within the SCSI device.

When a SCSI port detects an event, it shall use the event notification services (see 6.4) to notify SCSI application layer objects that the event has been detected.

The events detected and event notification services usage depends on whether the SCSI device is a SCSI target device (see figure 40) or a SCSI initiator device (see figure 41).

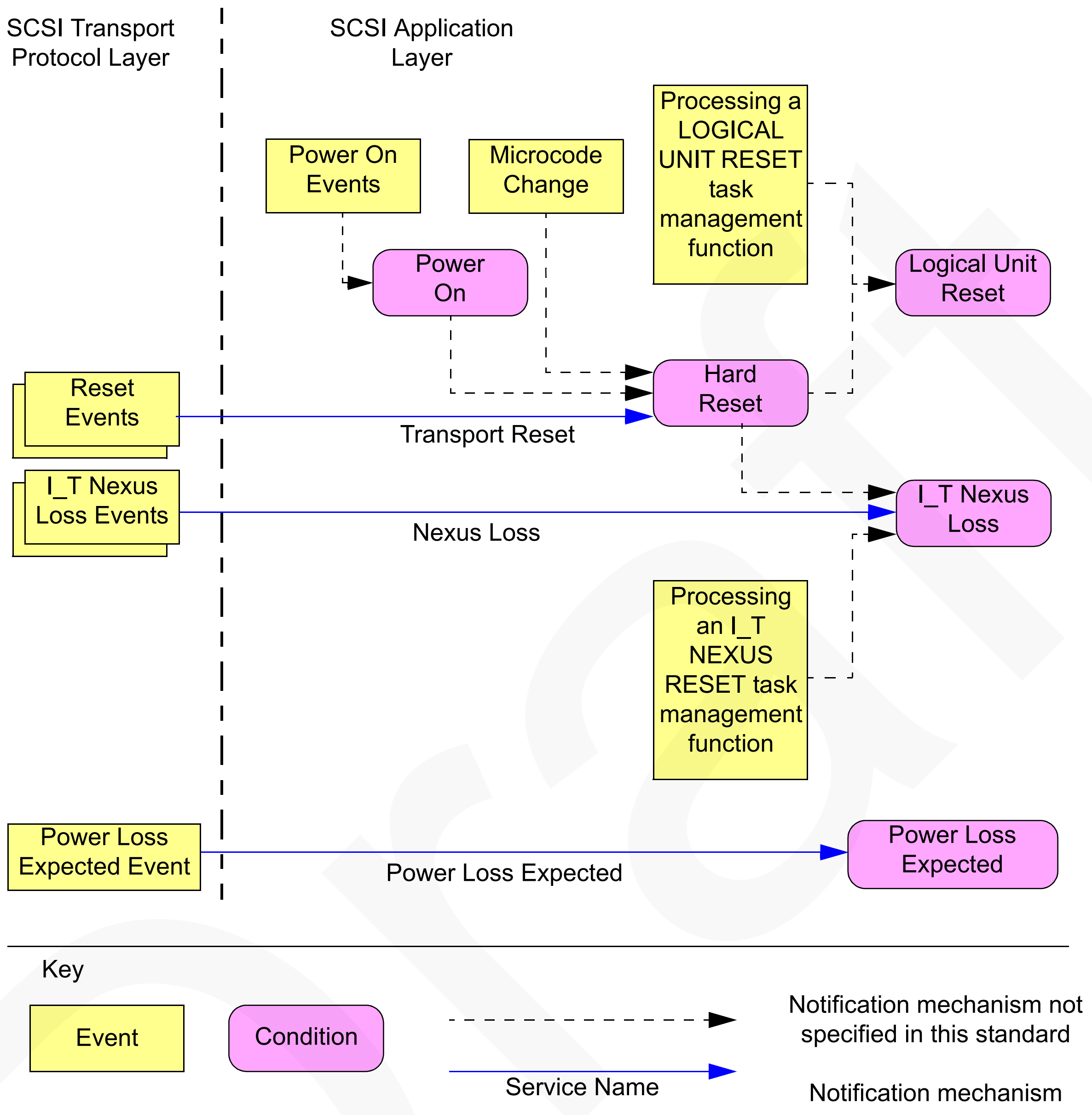


Figure 40 — Events and event notifications for SCSI target devices

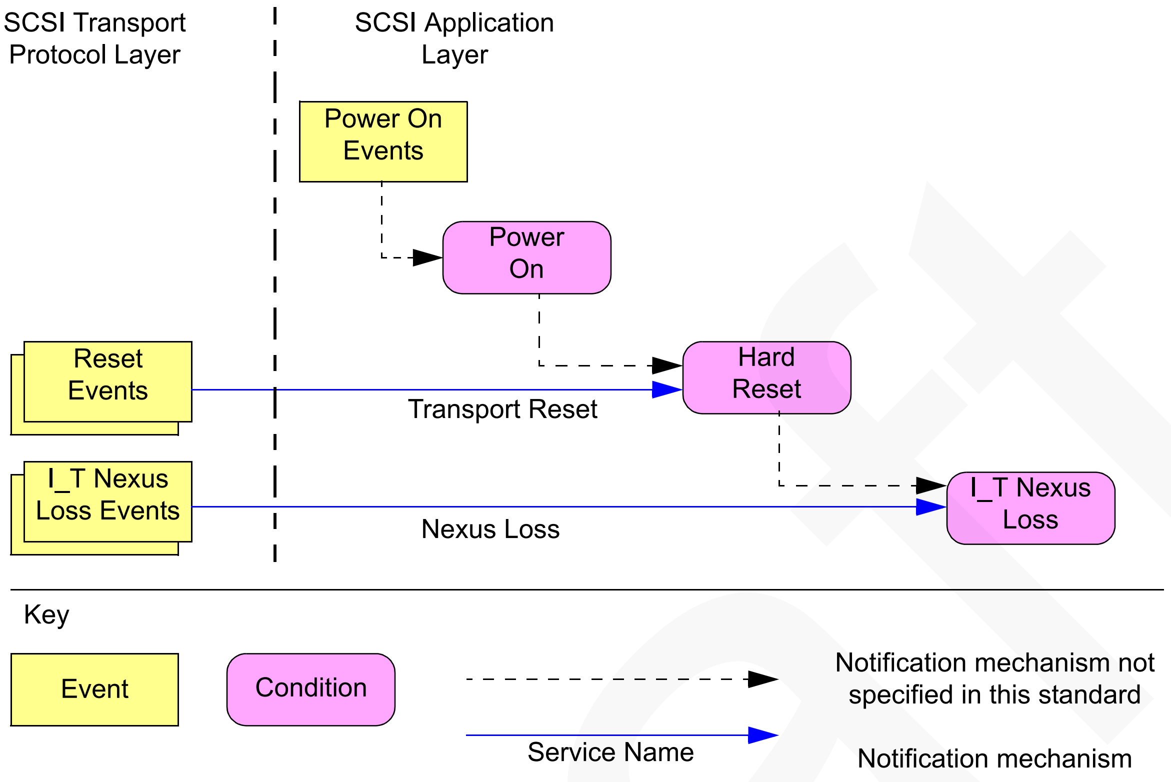


Figure 41 — Events and event notifications for SCSI initiator devices

6.2 Establishing a unit attention condition subsequent to detection of an event

Table 33 shows the additional sense code that a logical unit shall use when a unit attention (see 5.8.7) is established for each of the conditions shown in figure 40 (see 6.1). A SCSI transport protocol may define a more specific additional sense code than SCSI BUS RESET OCCURRED for reset events. The most specific condition in table 33 known to the logical unit should be used to establish the additional sense code for a unit attention.

Table 33 — Unit attention additional sense codes for events detected by SCSI target devices

Condition	Additional Sense Code	Specificity
Logical unit is unable to distinguish between the conditions	POWER ON, RESET, OR BUS DEVICE RESET OCCURRED	Lowest
Power loss expected	COMMANDS CLEARED BY POWER LOSS NOTIFICATION	
Power on	POWER ON OCCURRED or DEVICE INTERNAL RESET	
Hard reset	SCSI BUS RESET OCCURRED or MICROCODE HAS BEEN CHANGED or protocol specific	
Logical unit reset	BUS DEVICE RESET FUNCTION OCCURRED	
I_T nexus loss	I_T NEXUS LOSS OCCURRED	Highest

NOTE 10 - The names of the unit attention conditions listed in the subclause (e.g., SCSI BUS RESET OCCURRED) are based on usage in previous versions of this standard. The use of these unit attention condition names is not to be interpreted as a description of how the unit attention conditions are represented by any given SCSI transport protocol.

A logical unit may use the I_T NEXUS LOSS OCCURRED additional sense code when establishing a unit attention condition if:

a) The SCSI initiator port to which the sense data is being delivered is the SCSI initiator port that was associated with the I_T nexus loss, and the logical unit has maintained all state information specific to that SCSI initiator port since the I_T nexus loss; and
b) The I_T nexus being used to deliver the sense data is the same I_T nexus that was lost, and the logical unit has maintained all state information specific to that I_T nexus since the I_T nexus loss.

Otherwise, the logical unit shall use one of the less specific additional sense codes (e.g., POWER ON OCCURRED) when establishing a unit attention condition.

6.3 Conditions resulting from SCSI events

6.3.1 Power on

Power on is a SCSI device condition resulting from a power on event. When a SCSI device is powered on, it shall cause a hard reset.

The power on condition applies to both SCSI initiator devices and SCSI target devices.

6.3.2 Hard reset

Hard reset is a SCSI device condition resulting from:

a) A power on condition (see 6.3.1);
b) Microcode change (see SPC-4); or
c) A reset event indicated by a **Transport Reset** event notification (see 6.4).

The definition of reset events and the notification of their detection is SCSI transport protocol specific.

Each SCSI transport protocol standard that defines reset events shall specify a SCSI target port's protocol specific actions in response to reset events. Each SCSI transport protocol standard that defines reset events should specify when those events result in the delivery of a **Transport Reset** event notification to the SCSI applications layer.

SCSI transport protocols may include reset events that have no SCSI effects (e.g., a Fibre Channel non-initializing loop initialization primitive).

The hard reset condition applies to both SCSI initiator devices and SCSI target devices.

A SCSI target port's response to a hard reset condition shall include a logical unit reset condition (see 6.3.3) for all logical units to which the SCSI target port has access. A hard reset condition shall not affect any other SCSI target ports in the SCSI target device, however, the logical unit reset condition established by a hard reset may affect tasks that are communicating via other SCSI target ports.

Although the task manager response to task management requests is subject to the presence of access restrictions, as managed by ACCESS CONTROL OUT commands (see SPC-3), a hard reset condition shall not be prevented by access controls.

When a SCSI initiator port detects a hard reset condition, it should terminate all its outstanding **Execute Command** procedure calls with a service response of SERVICE DELIVERY OR TARGET FAILURE. A hard reset condition shall not affect any other SCSI initiator ports in the SCSI initiator device, however, the logical unit reset condition established in a SCSI target device by a hard reset may affect tasks that are communicating via other SCSI initiator ports.

A SCSI port's response to a hard reset condition shall include establishing an I_T nexus loss condition (see 6.3.4) for every I_T nexus associated with that SCSI port.

6.3.3 Logical unit reset

Logical unit reset is a logical unit condition resulting from:

a) A hard reset condition (see 6.3.2); or
b) A logical unit reset event indicating that a LOGICAL UNIT RESET task management request (see 7.7) has been processed.

The logical unit reset condition applies only to SCSI target devices.

When responding to a logical unit reset condition, the logical unit shall:

a) Abort all tasks as described in 5.6;
b) Clear all ACA conditions (see 5.8.2.5) in all task sets in the logical unit;
c) Establish a unit attention condition (see 5.8.7 and 6.2);
d) Initiate a logical unit reset for all dependent logical units (see 4.5.19.4); and
e) Perform any additional functions required by the applicable command standards.

6.3.4 I_T nexus loss

I_T nexus loss is a SCSI device condition resulting from:

a) A hard reset condition (see 6.3.2);
b) An I_T nexus loss event (e.g., logout) indicated by a **Nexus Loss** event notification (see 6.4); or
c) An I_T nexus loss event indicating that an I_T NEXUS RESET task management request (see 7.6) has been processed.

An I_T nexus loss event is an indication from the SCSI transport protocol to the SCSI application layer that an I_T nexus no longer exists. SCSI transport protocols may define I_T nexus loss events.

Each SCSI transport protocol standard that defines I_T nexus loss events should specify when those events result in the delivery of a **Nexus Loss** event notification to the SCSI applications layer.

The I_T nexus loss condition applies to both SCSI initiator devices and SCSI target devices.

When a SCSI target port detects an I_T nexus loss, a **Nexus Loss** event notification indication shall be delivered to each logical unit to which the I_T nexus has access. In response to the resulting I_T nexus loss condition a logical unit shall take the following actions:

a) Abort all tasks received on the I_T nexus as described in 5.6;
b) Clear all ACA conditions (see 5.8.2.5) associated with the I_T nexus;
c) Establish a unit attention condition for the SCSI initiator port associated with the I_T nexus (see 5.8.7 and 6.2); and
d) Perform any additional functions required by the applicable command standards.

If the logical unit retains state information for the I_T nexus that is lost, its response to the subsequent I_T nexus re-establishment for the logical unit should include establishing a unit attention with an additional sense code set to I_T NEXUS LOSS OCCURRED.

If the logical unit does not retain state information for the I_T nexus that is lost, it shall consider the subsequent I_T nexus re-establishment, if any, as the formation of a new I_T nexus for which there is no past history (e.g., establish a unit attention with an additional sense code set to POWER ON OCCURRED).

When a SCSI initiator port detects an I_T nexus loss, it should terminate all its outstanding **Execute Command** procedure calls and **Send Task Management Request** procedure calls for the SCSI target port associated with the I_T nexus with a service response of SERVICE DELIVERY OR TARGET FAILURE.

6.3.5 Power loss expected

Power loss expected is a SCSI device condition resulting from a power loss expected event indicated by a Power Loss Expected event notification (see 6.4).

A power loss expected event is an indication from the SCSI transport protocol to the SCSI application layer that power loss may occur within a protocol specific period of time. SCSI transport protocols may define power loss expected events.

Each SCSI transport protocol standard that defines power loss expected events should specify when those events result in the delivery of a Power Loss Expected event notification to the SCSI applications layer.

The power loss expected condition applies only to SCSI target devices and is equivalent to a CLEAR TASK SET task management function (see 7.5) applied to all task sets.

When a SCSI target port detects a power loss expected, a Power Loss Expected event notification indication shall be delivered to each logical unit to which the I_T nexus has access. In response to the resulting I_T power loss expected condition a logical unit shall take the following actions:

a) Abort all tasks and establish a unit attention condition as described in 5.6; and
b) Perform any additional functions required by the applicable protocol standards.

6.4 Event notification SCSI transport protocol services

The SCSI transport protocol services described in this subclause are used by a SCSI initiator port or a SCSI target port to deliver an indication to the SCSI application layer that a SCSI event has been detected.

All SCSI transport protocol standards should define the SCSI transport protocol specific requirements for implementing the **Nexus Loss** indication, the **Transport Reset** indication and the **Power Loss Expected** indication described in this subclause and when these indications are to be delivered to the SCSI applications layer.

The **Nexus Loss** indication and the **Transport Reset** indication are defined for both SCSI target devices and SCSI initiator devices.

Indication delivered to device servers, task managers, and application clients:

Nexus Loss (IN (I_T Nexus))

Argument description:

I_T Nexus: The specific I_T nexus that has been detected as lost.

Indication delivered to device servers, task managers, and application clients:

Transport Reset (IN (SCSI Port))

Argument descriptions:

SCSI Port: The specific SCSI port in the SCSI device for which a transport reset was detected.

The **Power Loss Expected** indication is defined for SCSI target devices.

Indication delivered to device servers and task managers.

Power Loss Expected (IN (SCSI Port))

Argument descriptions:

SCSI Port: The specific SCSI port in the SCSI device for which an unexpected power loss was detected.

7 Task management functions

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 34.

Table 34 — 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 tag nexus (see 4.7).

Output arguments:

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

One of the following SCSI transport protocol specific 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 3.1.112) 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.

7.2 ABORT TASK

Request:

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

Description:

This function shall be supported by all logical units.

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

A response of FUNCTION COMPLETE shall indicate that the task was aborted or was not in the task set. In either case, the SCSI target device shall guarantee that no further requests or responses are sent from the task.

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

7.3 ABORT TASK SET

Request:

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

Description:

This function shall be supported by all logical units.

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

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

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

7.4 CLEAR ACA

Request:

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

Description:

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

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

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

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

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

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

7.5 CLEAR TASK SET

Request:

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

Description:

This function shall be supported by logical units.

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

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

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

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

7.6 I_T NEXUS RESET

Request:

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

Description:

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

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

7.7 LOGICAL UNIT RESET

Request:

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

Description:

This function shall be supported by all logical units.

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

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

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

7.8 QUERY TASK

Request:

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

Description:

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

The task manager in the specified logical unit shall:

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

7.9 QUERY TASK SET

Request:

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

Description:

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

The task manager in the specified logical unit shall:

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

7.10 QUERY UNIT ATTENTION

Request:

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

Description:

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

The task manager in the specified logical unit shall:

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

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

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

Table 35 — Additional Response Information argument for QUERY UNIT ATTENTION

Bit / Byte	7	6	5	4	3	2	1	0
0	Reserved		UADE DEPTH		SENSE KEY			
1	ADDITIONAL SENSE CODE							
2	ADDITIONAL SENSE CODE QUALIFIER							

The UADE DEPTH field indicates the number of pending unit attention conditions or deferred errors and is defined in table 36.

Table 36 — UADE DEPTH field

Code	Description
00b	The combined number of unit attention conditions and deferred errors is unknown
01b	The combined number of unit attention conditions and deferred errors is one
10b	The combined number of unit attention conditions and deferred errors is greater than one
11b	Reserved

The SENSE KEY field indicates the value of the SENSE KEY field that would be returned in the sense data for the highest-priority pending unit attention condition or deferred error (see SPC-4).

The ADDITIONAL SENSE CODE field indicates the value of the ADDITIONAL SENSE CODE field in the highest-priority pending unit attention condition or deferred error (see SPC-4).

The ADDITIONAL SENSE CODE QUALIFIER field indicates the value of the ADDITIONAL SENSE CODE QUALIFIER field in the highest-priority pending unit attention condition or deferred error (see SPC-4).

7.11 Task management function lifetime

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

a) the task manager sends a SCSI transport protocol service response for the task management function;
b) an I_T nexus loss (see 6.3.4);
c) a logical unit reset (see 6.3.3);
d) a hard reset (see 6.3.2); or
e) a power on condition (see 6.3.1).

The application client maintains an application client task to interact with the task management function from the time the **Send Task Management Request** SCSI transport protocol service request is invoked until it receives one of the following SCSI target device responses:

a) A service response of FUNCTION COMPLETE, FUNCTION SUCCEEDED, FUNCTION REJECTED, or SERVICE DELIVERY OR TARGET FAILURE is received for that task management function;
b) Notification of a unit attention condition with any additional sense code whose ADDITIONAL SENSE CODE field contains 29h (e.g., POWER ON, RESET, OR BUS DEVICE RESET OCCURRED; POWER ON

OCCURRED; SCSI BUS RESET OCCURRED; BUS DEVICE RESET FUNCTION OCCURRED; DEVICE INTERNAL RESET; or I_T NEXUS LOSS OCCURRED);

c) Notification of a unit attention condition with an additional sense code of MICROCODE HAS BEEN CHANGED; or
d) Notification of a unit attention condition with an additional sense code of COMMANDS CLEARED BY POWER LOSS NOTIFICATION.

NOTE 12 - The names of the unit attention conditions listed in the subclause (e.g., SCSI BUS RESET OCCURRED) are based on usage in previous versions of this standard. The use of these unit attention condition names is not to be interpreted as a description of how the unit attention conditions are represented by any given SCSI transport protocol.

7.12 Task management SCSI transport protocol services

7.12.1 Task management SCSI transport protocol services overview

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

Nexus:	An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.7).
Function Identifier:	Argument encoding the task management function to be performed.

All SCSI transport protocol standards shall define the SCSI transport protocol specific requirements for implementing the **Send Task Management Request** request (see 7.12.2), the **Task Management Request Received** indication (see 7.12.3), the **Task Management Function Executed** response (see 7.12.4), and the **Received Task Management Function Executed** (see 7.12.5) confirmation SCSI transport protocol services.

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

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

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

7.12.2 Send Task Management Request transport protocol service request

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

Send Task Management Request transport protocol service request:

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

Input arguments:

Nexus:	An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.7).
Function Identifier:	Argument encoding the task management function to be performed.

7.12.3 Task Management Request Received transport protocol service indication

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

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

7.12.5 Received Task Management Function Executed transport protocol service confirmation

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

Received Task Management Function Executed transport protocol service confirmation:

Received Task Management Function Executed (IN (Nexus, Service Response))

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

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

7.13 Task management function example

Figure 42 shows the sequence of events associated with a task management function.

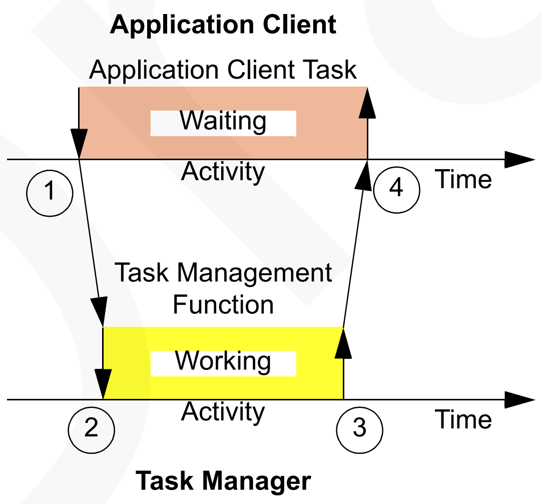


Figure 42 — Task management processing events

The numbers in figure 42 identify the events described as follows:

1) The application client task issues a task management request by invoking the **Send Task Management Request** SCSI transport protocol service.
2) The task manager is notified through a **Task Management Request Received** and begins processing the function.

3) The task manager performs the requested function and responds by invoking the **Task Management Function Executed** SCSI transport protocol service to notify the application client. The service response argument is set to a value of FUNCTION COMPLETE.
4) A **Received Task Management Function Executed** confirmation is received by the application client task.

8 Task set management

8.1 Introduction to task set management

This clause describes some of the controls that application clients have over task set management behaviors (see 8.3). This clause also specifies task set management requirements in terms of:

a) Task states (see 8.5);
b) Task attributes (see 8.6);
c) Task priority (see 8.7);
d) The events that cause transitions between task states (see 8.4 and 8.5); and
e) A map of task state transitions (see 8.8).

This clause concludes with several task set management examples (see 8.9).

Task behavior, as specified in this clause, refers to the functioning of a task as observed by an application client, including the results of command processing and interactions with other tasks.

The requirements for task set management only apply to a task after it has been entered into a task set. A task shall be entered into a task set unless:

a) A condition exists that causes that task to be completed with a status of BUSY, RESERVATION CONFLICT, TASK SET FULL, or ACA ACTIVE;
b) Detection of an overlapped command (see 5.8.3) causes that task to be completed with a CHECK CONDITION status; or
c) SCSI transport protocol specific errors cause that task to be completed with CHECK CONDITION status.

8.2 Implicit head of queue

A command standard (see 3.1.20) may define commands each of which may be processed by the task manager as if the task's task attribute is HEAD OF QUEUE even if the task is received with a SIMPLE task attribute, an ORDERED task attribute, or no task attribute.

An application client should not send a command with the ORDERED task attribute if the command may be processed as if it has a task attribute of HEAD OF QUEUE because whether the ORDERED task attribute is honored is vendor specific.

8.3 Task management model

The task management model requires the following task set management behaviors:

a) The SIMPLE task attribute (see 8.6.1) shall be supported;
b) Task attributes other than SIMPLE may be supported;
c) The QUEUE ALGORITHM MODIFIER field in the Control mode page (see SPC-3) shall control the processing sequence of tasks having the SIMPLE task attribute;
d) The QERR field in the Control mode page (see SPC-3) shall control aborting of tasks when a CHECK CONDITION status is returned for any task; and
e) The CLEAR TASK SET task management function (see 7.5) shall be supported.

8.4 Task management events

The following describe the events that cause changes in task state.

All older tasks ended:	If the TST field in the Control mode page (see SPC-3) equals 000b, all tasks received on all I_T nexuses and accepted earlier in time than the referenced task have ended. If the TST field equals 001b, all tasks received on the referenced I_T nexus and accepted earlier in time than the referenced task have ended.
All head of queue and older ordered tasks ended:	If the TST field equals 000b, all the following tasks received on all I_T nexuses have ended: a) All head of queue tasks; and b) All ordered tasks accepted earlier in time than the referenced task. If the TST field equals 001b, the following tasks received on the referenced I_T nexus have ended: a) All head of queue tasks; and b) All ordered tasks accepted earlier in time than the referenced task.
ACA establishment:	An ACA condition has been established (see 5.8.1).
task abort:	A task has been aborted as described in 5.6.
task completion:	The device server has sent a service response of TASK COMPLETE for the task (see 5.1 and 5.5).
task ended:	A task has completed or aborted.
ACA cleared:	An ACA condition has been cleared (see 5.8.2.5).

8.5 Task states

8.5.1 Overview

8.5.1.1 Task state nomenclature

This standard defines four tasks states, summarized in table 37.

Table 37 — Task State Nomenclature

Task State Name	Reference	Tasks in This State May Be Called
Enabled task state	8.5.2	Enabled tasks
Blocked task state	8.5.3	Blocked tasks
Dormant task state	8.5.4	Dormant tasks
Ended task state	8.5.5	Ended tasks

8.5.1.2 Suspended information

Any information the logical unit has or accepts for a task in the blocked task state (see 8.5.3) or dormant task state (see 8.5.4) is required to be held in a condition where it is not available to the task. Such information is called suspended information.

8.5.2 Enabled task state

A task in the enabled task state may become a current task and may complete at any time, subject to the task completion constraints specified in the Control mode page (see SPC-3). A task that has been accepted into the task set shall not complete or become a current task unless it is in the enabled task state.

Except for the use of resources required to preserve task state, a task shall produce no effects detectable by the application client before the task's first transition to the enabled task state. Before entering this state for the first time, the task may perform other activities visible at the STPL (e.g., pre-fetching data to be written to the media), however this activity shall not result in a detectable change in state as perceived by an application client. In addition, the behavior of a completed task, as defined by the commands it has processed, shall not be affected by the task's states before it enters the enabled task state.

8.5.3 Blocked task state

A task in the blocked task state is prevented from completing due to an ACA condition. A task in this state shall not become a current task. While a task is in the blocked task state, any information the logical unit has or accepts for the task shall be suspended. If the TST field in the Control mode page (see SPC-3) equals 000b the blocked task state is independent of I_T nexus. If the TST field equals 001b the blocked task state applies only to the faulted I_T nexus.

8.5.4 Dormant task state

A task in the dormant task state is prevented from completing due to the presence of certain other tasks in the task set. A task in this state shall not become a current task. While a task is in the dormant task state, any information the logical unit has or accepts for the task shall be suspended.

8.5.5 Ended task state

A task in the ended task state is removed from the task set.

8.5.6 Task states and task lifetimes

Figure 43 shows the events corresponding to two task processing sequences. Except for the dormant task state between times A and B in case 1, logical unit conditions and the commands processed by the task are identical. Assuming in each case the task completes with a status of GOOD at time C, the state observed by the application client for case 1 shall be indistinguishable from the state observed for case 2.

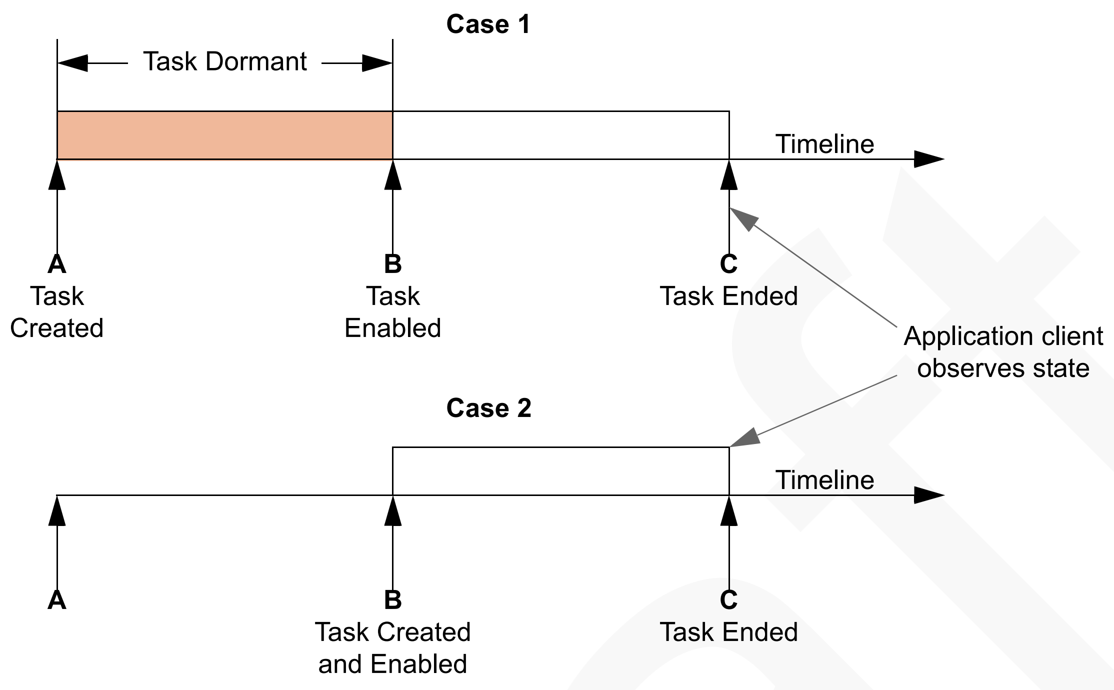


Figure 43 — Example of Dormant state task behavior

8.6 Task attributes

8.6.1 Overview

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

Table 38 — Task attributes

Task Attribute	Reference
SIMPLE	8.6.2
ORDERED	8.6.3
HEAD OF QUEUE	8.6.4
ACA	8.6.5

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

8.6.2 Simple task

If accepted, a task having the SIMPLE task attribute shall be entered into the task set in the dormant task state. The task shall not enter the enabled task state until all head of queue tasks and older ordered tasks in the task set have ended (see 8.4).

The QUEUE ALGORITHM MODIFIER field in the Control mode page (see SPC-3) provides additional constraints on task completion order for tasks having the SIMPLE task attribute.

8.6.3 Ordered task

If accepted, a task having the ORDERED task attribute shall be entered into the task set in the dormant task state. The task shall not enter the enabled task state until all head of queue tasks and all older tasks in the task set have ended (see 8.4).

8.6.4 Head of queue task

If accepted, a task having the HEAD OF QUEUE task attribute shall be entered into the task set in the enabled task state.

8.6.5 ACA task

If accepted, a task having the ACA task attribute shall be entered into the task set in the enabled task state. There shall be no more than one ACA task per task set (see 5.8.2.2).

8.7 Task priority

Task priority specifies the relative scheduling importance of a task having a SIMPLE task attribute in relation to other tasks having SIMPLE task attributes already in the task set. If the task has a task attribute other than SIMPLE, the task priority is not used. Task priority is a value in the range of 0h through Fh. A task with either no task priority or a task priority set to 0h has a vendor-specific level of scheduling importance. A task with a task priority set to 1h has the highest scheduling importance, with increasing task priority values indicating decreasing scheduling importance. A task with a task priority set to Fh has the lowest scheduling importance.

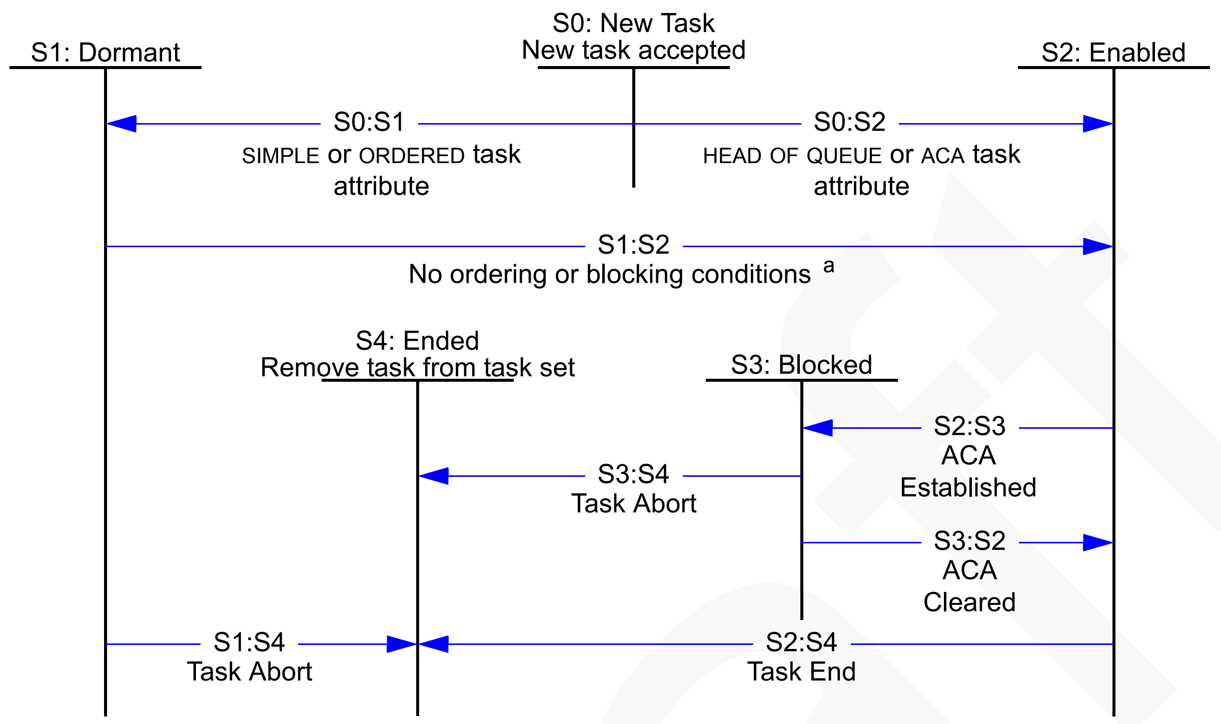
If the Task Priority argument is set to zero or is not contained within the Send SCSI Received SCSI transport protocol service indication (see 5.4.2) and a priority has been assigned to the I_T_L nexus, the device server shall use that priority as the task priority. A priority may be assigned to an I_T_L nexus by a SET PRIORITY command (see SPC-3) or by the INITIAL PRIORITY field in the Control Extension mode page (see SPC-3). If no priority has been assigned to the I_T_L nexus using the SET PRIORITY command and the logical unit does not support the INITIAL PRIORITY field in the Control Extension mode page the device server shall set the task priority to 0h (i.e., vendor specific) or the task shall have no task priority.

A task manager may use task priority to determine an ordering to process tasks with the SIMPLE task attribute within the task set. A difference in task priority between tasks may not override other scheduling considerations (e.g., different times to access different logical block addresses) or vendor specific scheduling considerations. However, processing of a collection of tasks with different task priorities should cause the subset of tasks with the higher task priorities to return status sooner in aggregate than the same subset would if the same collection of tasks were submitted under the same conditions but with all task priorities being equal.

8.8 Task state transitions

This subclause describes task state transitions, actions and associated triggering events as they appear to an application client. The logical unit response to events affecting multiple tasks (e.g., a CLEAR TASK SET) may be different from the response to an event affecting a single task. To the application client, the collective behavior appears as a series of state changes occurring to individual tasks.

The task state diagram of figure 44 shows the behavior of a single task in response to an external event.



a ACA is not active and:
a) For simple tasks, all head of queue and all older ordered tasks have ended; or
b) For ordered tasks, all head of queue tasks and all older tasks have ended.

Figure 44 — Task states

Transition S0:S1: If a newly accepted task has the SIMPLE or ORDERED task attribute, it shall transition to the dormant task state.

Transition S0:S2: If a newly accepted task has the HEAD OF QUEUE or ACA task attribute, it shall transition to the enabled task state.

Transition S1:S2: The task attribute of a dormant task shall affect the transition to the enabled task state as follows:

a) A dormant task having the SIMPLE task attribute shall enter the enabled task state when all head of queue and older ordered tasks (see 8.4) have ended; or
b) A dormant task having the ORDERED task attribute shall enter the enabled task state when all head of queue tasks and all older tasks (see 8.4) have ended.

If the TST field in the Control mode page (see SPC-3) contains 000b, then the transition from dormant task to enabled task shall not occur while an ACA is in effect for any I_T nexus (see 5.8.2.3 and 5.8.2.4). If the TST field contains 001b, then dormant tasks from the faulted I_T nexus shall not transition to the enabled task state while an ACA is in effect for that I_T nexus (see 5.8.2.3).

Transition S2:S3: The establishment of an ACA condition (see 8.4) shall cause zero or more enabled tasks to enter the blocked task state as described in 5.8.2.2.

Transition S3:S2: When an ACA condition is cleared (see 8.4), tasks that entered the blocked state when the ACA condition was established (see 5.8.2.2) shall re-enter the enabled task state.

Transition S2:S4: A task that has completed (see 8.4) or aborted (see 8.4 and 5.6) shall enter the ended task state. This is the only state transition out of S2:Enabled that applies to an ACA task.

Transitions S1:S4, S3:S4: A task abort event (see 8.4 and 5.6) shall cause the task to unconditionally enter the ended task state.

8.9 Task set management examples

8.9.1 Introduction

Several task set management scenarios are shown in 8.9.2, 8.9.3, and 8.9.4. The examples are valid for configurations with one or multiple SCSI initiator ports when the TST field contains 000b (i.e., the interaction among tasks in a task set is independent of the I_T nexus on which a task is received). The examples are also valid for a single I_T nexus when the TST field contains 001b (i.e., task set management proceeds independently for each I_T nexus and the events and transitions for the task set associated with one I_T nexus do not affect the task set management for task sets associated with other I_T nexuses). Throughout these examples, the scope of the task set box drawn in each snapshot depends on the setting of the TST field in the Control mode page (see SPC-3).

The figure accompanying each example shows successive snapshots of a task set after various events (e.g., task creation or completion). In all cases, the constraints on task completion order established using Control mode page (see SPC-3) fields other that the TST field (e.g., the QUEUE ALGORITHM MODIFIER field) are not in effect.

A task set is shown as an ordered list or queue of tasks with the head of the queue towards the top of the figure. A new head of queue task always enters the task set at the head, displacing older head of queue tasks. Simple, ordered and ACA tasks always enter the task set at the end of the queue.

Tasks, denoted by rectangles, are numbered in ascending order from oldest to most recent. Fill, shape and line weight are used to distinguish task states and task attributes are shown in table 39.

Table 39 — Task attribute and state indications in examples

Task Attribute	Box Shape	Line Weight	Task State
SIMPLE	Rounded Corners	Thin	Enabled
ORDERED	Square Corners	Thin	Dormant
HEAD OF QUEUE	Square Corners	Thick	Blocked
ACA	Square Corners	Thin Dashed	

The conditions preventing a dormant task from entering enabled task state, except for ACA conditions, are shown by means of blocking boundaries. Such boundaries appear as horizontal lines with an arrow on both ends. The tasks causing the barrier condition are described as part of each example. A task is impeded by the barrier if it is between the boundary and the end of the queue. When ACA is not in effect, a task may enter the enabled task state after all intervening barriers have been removed.

8.9.2 Head of queue tasks

Figure 45 shows task set conditions when several head of queue tasks are processed.

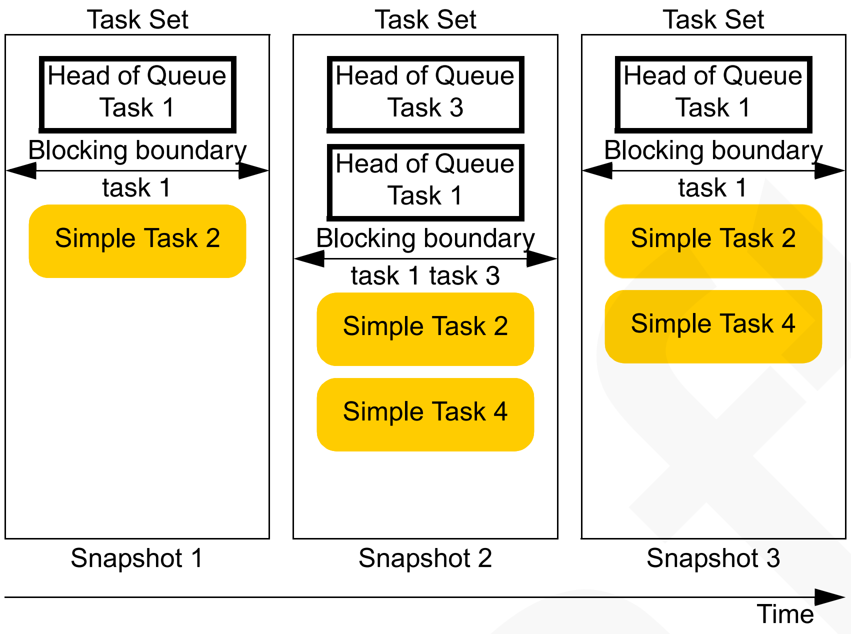


Figure 45 — Head of queue tasks and blocking boundaries (example 1)

In snapshot 1 the task set contains one head of queue and one simple task. As shown by the blocking boundary, simple task 2 is in the dormant task state because of the head of queue task. Snapshot 2 shows the task set after head of queue task 3 and simple task 4 are created. The new head of queue task is placed at the front of the queue in the enabled task state, displacing task 1. Snapshot 3 shows the task set after task 3 completes. Since the conditions indicated by the task 1 blocking boundary are still in effect, task 2 and task 4 remain in the dormant task state.

Figure 46 is the same as the previous example, except that task 1 completes instead of task 3.

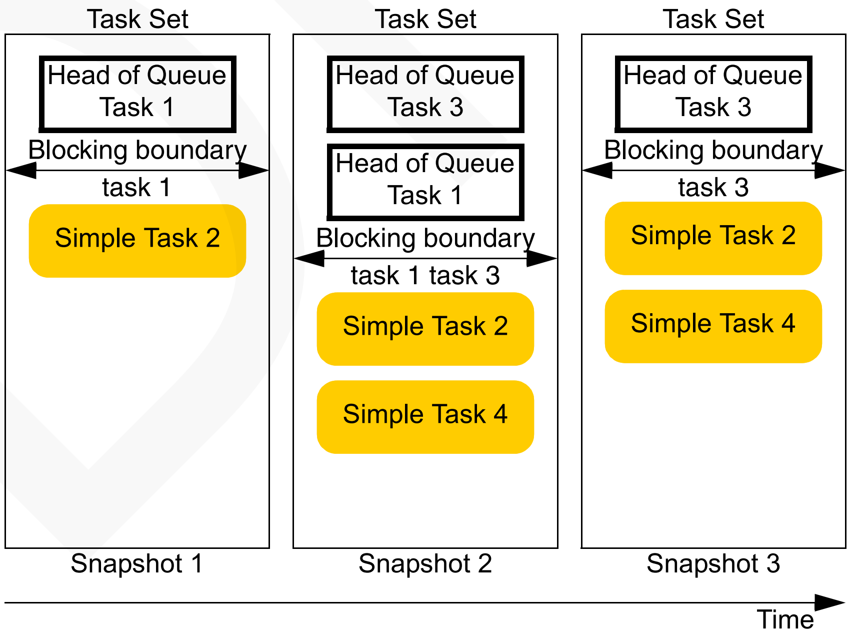


Figure 46 — Head of queue tasks and blocking boundaries (example 2)

Because the blocking boundary remains in place for a head of queue task, both simple tasks remain in the dormant task state in snapshot 3. The blocking boundary is not removed until all head of queue tasks complete.

8.9.3 Ordered tasks

An example of ordered and simple task interaction is shown in figure 47.

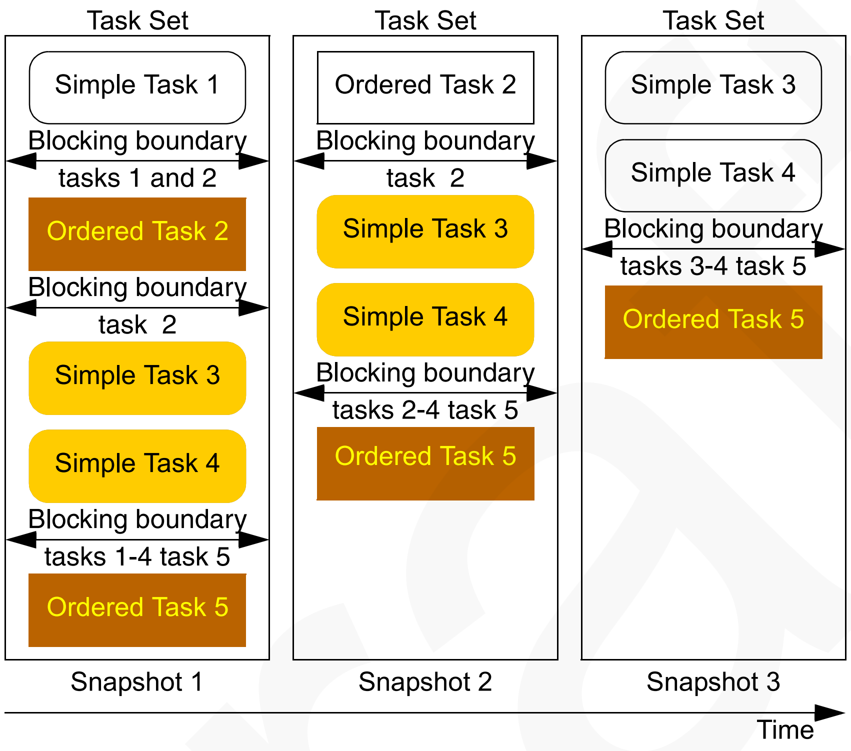


Figure 47 — Ordered tasks and blocking boundaries

The state of dormant task 2 through task 5 is determined by the requirements shown in table 40.

Table 40 — Dormant task blocking boundary requirements

Task	Reason for blocking boundary
2	An ordered task is not allowed to enter the enabled task state until all head of queue tasks and all older tasks have ended.
5	
3	A simple task is not allowed to enter the enabled task state until all head of queue and all older ordered tasks have ended.
4	

The table 40 constraints are shown by the blocking boundaries in snapshot 1.

In snapshot 2, the completion of task 1 allows ordered task 2 to enter the enabled task state. Since the initial constraints on task 3, task 4 and task 5 are still in effect, these tasks are required to remain in the dormant task state. As shown in snapshot 3, the completion of task 2 triggers two state changes, with task 3 and task 4 transitioning to the enabled task state. Task 5 is required to remain in the dormant task state until task 3 and task 4 complete.

8.9.4 ACA task

Figure 48 shows the effects of an ACA condition on the task set. This example assumes the QERR field contains 00b in the Control mode page (see SPC-3). Consequently, clearing an ACA condition does not cause tasks to be aborted

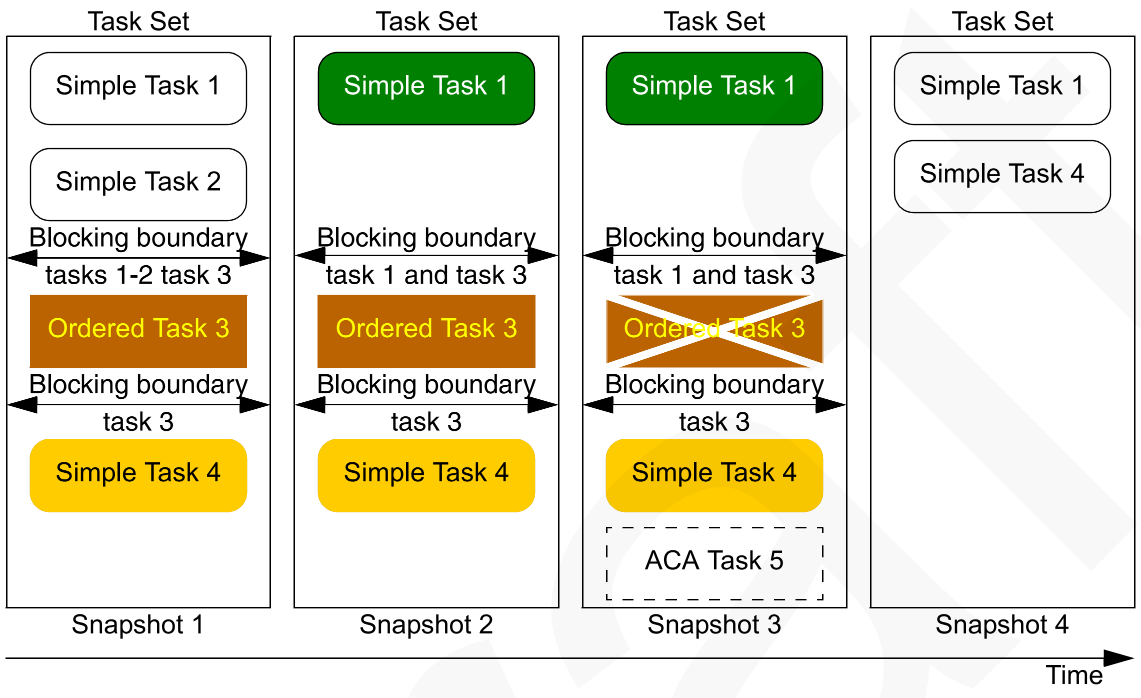


Figure 48 — ACA task example

The completion of task 2 with CHECK CONDITION status causes task 1 to enter the blocked task state shown in snapshot 2. In snapshot 3, ordered task 3 is aborted using the ABORT TASK task management function and ACA task 5 is created to perform additional handling for the exception. Once the ACA condition is cleared (i.e., snapshot 4), simple task 1 is allowed to reenter the enabled task state. Since there are no head of queue tasks or older ordered tasks, task 4 also transitions to the enabled task state.

Annex A

(informative)

Identifiers and names for objects

A.1 Identifiers and names overview

This annex summarizes identifiers and names.

The following SCSI architecture model objects have identifiers and names summarized in this annex:

a) SCSI initiator port (see 3.1.97);
b) SCSI target port (see 3.1.101);
c) Logical unit (see 3.1.62);
d) SCSI initiator device (see 3.1.96); and
e) SCSI target device (see 3.1.100).

A.2 Identifiers and names

This standard defines the identifiers and names for the objects listed in A.1. The size requirements placed on identifiers by this standard are as shown in table A.1. This standard places no requirements on the sizes of names. Table A.1 also lists whether this standard or SPC-4 requires SCSI transport protocols and logical units to support identifiers and names for an object.

Table A.1 — This standard and SPC-4 object size and support requirements

Object	Identifier		Name	
	Size	Support Requirements	Size	Support Requirements
Initiator device	n/a	n/a	not specified	optional
Target device	n/a	n/a	not specified [a]	optional
Initiator port	not specified	mandatory	not specified	optional
Target port	not specified	mandatory	not specified [a]	optional
Logical unit	8 bytes (maximum)	mandatory	not specified [a]	mandatory

[a] Reported in the Device Identification VPD page (see SPC-4).

Each SCSI transport protocol defines the size and format of identifiers and names for each object.

See table A.2 for a list of the size of the identifiers for each SCSI transport protocol.

Table A.2 — Object identifier size for each SCSI transport protocol

Object	Identifier size					
	FCP-2	SRP	iSCSI	SBP-3	SAS SSP	ADT
Initiator port	3 bytes	16 bytes	241 bytes [a]	2 bytes	8 bytes	none
Target port	3 bytes	16 bytes	233 bytes [a]	11 bytes	8 bytes	none
Logical unit	8 bytes	8 bytes	8 bytes	2 bytes	8 bytes	2 bytes
[a] Maximum size, including the terminating null character byte.						

See table A.3 for a list of the format of the identifiers for each SCSI transport protocol.

Table A.3 — Object identifier format for each SCSI transport protocol

Object	Identifier format					
	FCP-2	SRP	iSCSI [b]	SBP-3	SAS SSP	ADT
Initiator port	Fibre Channel address identifier	EUI-64 || 8 byte extension [a]	iSCSI name [c] || ",i,0x" || Initiator Session Identifier [d]	binary value	NAA IEEE Registered format	None
Target port	Fibre Channel address identifier	EUI-64 || 8 byte extension [a]	iSCSI name [c] || ",t,0x" || Target Portal Group Tag [e]	EUI-64 || Discovery ID [f]	NAA IEEE Registered format	None
Logical unit	as specified in this standard (see 4.6)	as specified in this standard (see 4.6)	as specified in this standard (see 4.6)	binary value (2 bytes)	as specified in this standard (see 4.6)	None

Key:

Key:	||	means "concatenated with"
	",i,0x"	means a UTF-8 string composed of the following five characters: comma, lowercase i, comma, zero, and lowercase x.
	",t,0x"	means a UTF-8 string composed of the following five characters: comma, lowercase t, comma, zero, and lowercase x.

[a] Required to be worldwide unique and recommend to be EUI-64 concatenated with an 8 byte extension.
[b] iSCSI identifiers are concatenated strings containing no null characters except after the last string in the concatenation.
[c] The iSCSI name portion of the string is a worldwide unique UTF-8 string no more than 223 bytes long, not including null character termination.
[d] The Initiator Session Identifier (ISID) portion of the string is a UTF-8 encoded hexadecimal representation of a six byte binary value. This portion of the string contains no more than 12 bytes, not including null character termination if any.
[e] The Target Portal Group Tag (TPGT) portion of the string is a UTF-8 encoded hexadecimal representation of a two byte binary value. This portion of the string contains no more than 4 bytes, not including null character termination if any.
[f] See ISO/IEC 13213:1994 for more information on the Discovery ID.

See table A.4 for a list of the size of the names for each SCSI transport protocol.

Table A.4 — Object name size for each SCSI transport protocol

Object	Name size [a]					
	FCP-2	SRP	iSCSI [b]	SBP-3	SAS SSP	ADT
Initiator device	not specified	not specified	224 bytes	not specified	8 bytes	not specified
Target device	not specified	not specified	224 bytes	not specified	8 bytes	not specified
Initiator port	8 bytes	16 bytes	241 bytes	8 bytes	not specified	not specified
Target port	8 bytes	16 bytes	233 bytes	11 bytes	not specified	not specified
Logical unit	Reported in the Device Identification VPD page (see SPC-3).					

[a] Any SCSI transport protocol may support the SCSI name string format (see SPC-3), resulting in names with the sizes shown in the iSCSI column.
[b] Maximum size, including the terminating null character byte.

See table A.5 for a list of the format of the names for each SCSI transport protocol.

Table A.5 — Object name format for each SCSI transport protocol

Object	Name format [a]					
	FCP-2	SRP	iSCSI [b]	SBP-3	SAS SSP	ADT
Initiator device	not specified	not specified	SCSI name string format	not specified	NAA IEEE Registered format	not specified
Target device	not specified	not specified	SCSI name string format	not specified	NAA IEEE Registered format	not specified
Initiator port	Fibre Channel name_ identifier	EUI-64 || 8 byte extension [d]	iSCSI name [c] || ",i,0x" || Initiator Session Identifier [e]	EUI-64	not specified	not specified
Target port	Fibre Channel name_ identifier	EUI-64 || 8 byte extension [d]	iSCSI name [c] || ",t,0x" || Target Portal Group Tag [f]	EUI-64 || Discovery ID [g]	not specified	not specified
Logical unit	Device Identification VPD page name (see SPC-3)					

Key:	||	means “concatenated with”
	",i,0x"	means a UTF-8 string composed of the following five characters: comma, lowercase i, comma, zero, and lowercase x.
	",t,0x"	means a UTF-8 string composed of the following five characters: comma, lowercase t, comma, zero, and lowercase x.

[a] In addition to the name formats shown in this table, any SCSI transport protocol may support the SCSI name string format (see SPC-3).
[b] iSCSI identifiers are concatenated strings containing no null characters except after the last string in the concatenation.
[c] The iSCSI name portion of the string is a worldwide unique UTF-8 string no more than 223 bytes long, not including null character termination.
[d] Required to be worldwide unique and recommend to be EUI-64 concatenated with an 8 byte extension.
[e] The Initiator Session Identifier (ISID) portion of the string is a UTF-8 encoded hexadecimal representation of a six byte binary value. This portion of the string contains no more than 12 bytes, not including null character termination if any.
[f] The Target Portal Group Tag (TPGT) portion of the string is a UTF-8 encoded hexadecimal representation of a two byte binary value. This portion of the string contains no more than 4 bytes, not including null character termination if any.
[g] See ISO/IEC 13213:1994 for more information on the Discovery ID.

A.3 SCSI transport protocol acronyms and bibliography

A.3.1 EUI-64 (Extended Unique Identifier, a 64-bit globally unique identifier): The IEEE maintains a tutorial describing EUI-64 at http://standards.ieee.org/regauth/oui/tutorials/EUI64.html.

A.3.2 FCP-2: SCSI Fibre Channel Protocol -2 (see 1.3).

A.3.3 ISO/IEC 13213:1994: See ISO/IEC 13213:1994, Information technology - Microprocessor systems - Control and Status Registers Architecture for microcomputer buses [ANSI/IEEE 1212, 1994 Edition]. See http://www.iso.org/.

A.3.4 iSCSI: internet SCSI (see RFC 3720, http://www.ietf.org/rfc/rfc3720.txt).

A.3.5 NAA: Name Address Authority (see SPC-3).

A.3.6 SAS: Serial Attached SCSI (see 1.3).

A.3.7 SAS SSP: SAS (see A.3.6) Serial SCSI Protocol.

A.3.8 SBP-3: Serial Bus Protocol -3 (see 1.3).

A.3.9 SRP: SCSI RDMA Protocol (see 1.3).

A.3.10 UTF-8: See ISO/IEC 10646-1:2000, Information technology - Universal Multiple-Octet Coded Character Set (UCS) - Part 1: Architecture and Basic Multilingual Plane. See http://www.iso.org/.