

INTERNATIONAL STANDARD

ISO/IEC 14776-413

First edition
2007-02

Information technology – Small computer system interface (SCSI) – Part 413: Architecture model-3 (SAM-3)



Reference number
ISO/IEC 14776-413:2007(E)

AVA Documents
Document Sharing Hub

Document Reference: ISO 14776-413-2007

INTERNATIONAL STANDARD

ISO/IEC 14776-413

First edition
2007-02

Information technology – Small computer system interface (SCSI) – Part 413: Architecture model-3 (SAM-3)

Copyright © 2007 ISO/IEC, Geneva — All rights reserved

No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland
Telephone: +41 22 919 02 11 Telefax: +41 22 919 03 00 E-mail: inmail@iec.ch Web: www.iec.ch



PRICE CODE **XA**

For price, see current catalogue

AVA Documents
Document Sharing Hub

Contents

	Page
Foreword	7
Introduction	8
1 General	10
1.1 Scope	10
1.2 Precedence requirements	10
2 Normative references.....	12
3 Definitions, symbols, abbreviations, and conventions	13
3.1 Definitions.....	13
3.2 Acronyms.....	21
3.3 Keywords.....	22
3.4 Editorial conventions	23
3.5 Numeric conventions	23
3.6 Notation conventions.....	24
3.6.1 Hierarchy diagram conventions.....	24
3.6.2 Notation for procedure calls.....	26
3.6.3 Notation for state diagrams	27
4 SCSI architecture model	28
4.1 Introduction.....	28
4.2 The SCSI distributed service model	28
4.3 The SCSI client - server model.....	29
4.4 The SCSI structural model	31
4.5 SCSI domain	32
4.6 The service delivery subsystem	33
4.6.1 The service delivery subsystem object.....	33
4.6.2 Synchronizing client and server states.....	33
4.6.3 Request/Response ordering.....	33
4.7 SCSI devices	34
4.7.1 General.....	34
4.7.2 SCSI initiator device	34
4.7.3 SCSI target device.....	35
4.7.4 SCSI target/initiator device	37
4.7.5 SCSI port identifier	38
4.7.6 Relative port identifier.....	38
4.7.7 SCSI task router	38
4.7.8 SCSI device name.....	38
4.7.9 SCSI port name	39
4.8 Logical units.....	39
4.9 Logical unit numbers	40
4.9.1 Introduction.....	40
4.9.2 Logical unit numbers overview	40
4.9.3 Minimum LUN addressing requirements	40
4.9.4 Single level logical unit number structure	41
4.9.5 Eight byte logical unit number structure	42
4.9.6 Peripheral device addressing method	44
4.9.7 Flat space addressing method	45
4.9.8 Logical unit addressing method.....	45
4.9.9 Extended logical unit addressing.....	45
4.9.10 Well-known logical unit addressing	47
4.9.11 Logical unit not specified addressing.....	48
4.10 Well-known logical units	48
4.11 Tasks and task tags.....	49
4.12 The nexus object	49
4.13 SCSI ports	49

4.13.1 SCSI port configurations.....	50
4.13.2 SCSI devices with multiple ports	50
4.13.3 Multiple port SCSI target device structure	51
4.13.4 Multiple port SCSI initiator device structure.....	52
4.13.5 Multiple port SCSI target/initiator device structure	53
4.13.6 SCSI initiator device view of a multiple port SCSI target device	53
4.13.7 SCSI target device view of a multiple port SCSI initiator device	56
4.14 Model for dependent logical units.....	57
4.15 The SCSI model for distributed communication	59
5 SCSI command model	64
5.1 The Execute Command procedure call	64
5.2 Command descriptor block (CDB).....	65
5.3 Status	67
5.3.1 Status codes.....	67
5.3.2 Status precedence.....	69
5.4 SCSI transport protocol services in support of Execute Command.....	69
5.4.1 Overview.....	69
5.4.2 Execute Command request/confirmation SCSI transport protocol services.....	69
5.4.3 Data transfer SCSI transport protocol services	72
5.4.3.1 Introduction.....	72
5.4.3.2 Data-In delivery service	73
5.4.3.3 Data-Out delivery service	74
5.4.3.4 Terminate Data Transfer service	74
5.5 Task and command lifetimes.....	75
5.6 Task management function lifetime.....	76
5.7 Aborting tasks.....	77
5.7.1 Mechanisms that cause tasks to be aborted	77
5.7.2 When a SCSI initiator port aborts tasks received on its own I_T nexus	77
5.7.3 When a SCSI initiator port aborts tasks received on other I_T nexuses	78
5.8 Command processing examples	78
5.8.1 Unlinked command example	78
5.8.2 Linked command example.....	79
5.9 Command processing considerations and exception conditions.....	80
5.9.1 Commands that complete with CHECK CONDITION status.....	80
5.9.1.1 Overview.....	80
5.9.1.2 Handling tasks when ACA is not in effect.....	80
5.9.1.3 Aborting other tasks when CHECK CONDITION status is returned without establishing an ACA.....	80
5.9.2 Auto contingent allegiance (ACA).....	80
5.9.2.1 ACA Overview	80
5.9.2.2 Establishing an ACA.....	81
5.9.2.3 Handling new tasks received on the faulted I_T nexus when ACA is in effect.....	81
5.9.2.4 Handling new tasks received on non-faulted I_T nexuses when ACA is in effect.....	82
5.9.2.4.1 Command processing permitted for tasks received on non-faulted I_T nexuses during ACA	82
5.9.2.4.2 Handling new tasks received on non-faulted I_T nexuses when ACA is in effect.....	83
5.9.2.5 Clearing an ACA condition	83
5.9.3 Overlapped commands	84
5.9.4 Incorrect logical unit selection	84
5.9.5 Task attribute exception conditions	84
5.9.6 Sense data	85
5.9.7 Unit Attention condition.....	85
6 SCSI events and event notification model	87
6.1 SCSI events overview	87
6.2 Establishing a unit attention condition subsequent to detection of an event	88
6.3 Conditions resulting from SCSI events.....	89
6.3.1 Power on	89
6.3.2 Hard reset.....	89
6.3.3 Logical unit reset	90
6.3.4 I_T nexus loss	90

6.4 Event notification SCSI transport protocol services.....	91
7 Task management functions.....	92
7.1 Introduction.....	92
7.2 ABORT TASK.....	93
7.3 ABORT TASK SET.....	93
7.4 CLEAR ACA.....	94
7.5 CLEAR TASK SET.....	94
7.6 LOGICAL UNIT RESET.....	95
7.7 QUERY TASK.....	95
7.8 Task management SCSI transport protocol services.....	95
7.9 Task management function example.....	97
8 Task set management.....	99
8.1 Introduction to task set management.....	99
8.2 Implicit head of queue.....	99
8.3 Task management models.....	99
8.3.1 Task management model management features.....	99
8.3.2 Full task management model.....	99
8.3.3 Basic task management model.....	100
8.4 Task management events.....	100
8.5 Task states.....	101
8.5.1 Overview.....	101
8.5.1.1 Task state nomenclature.....	101
8.5.1.2 Suspended information.....	101
8.5.2 Enabled task state.....	101
8.5.3 Blocked task state.....	101
8.5.4 Dormant task state.....	101
8.5.5 Ended task state.....	101
8.5.6 Task states and task lifetimes.....	102
8.6 Task attributes.....	103
8.6.1 Overview.....	103
8.6.2 Simple task.....	103
8.6.3 Ordered task.....	103
8.6.4 Head of queue task.....	103
8.6.5 ACA task.....	103
8.7 Task priority.....	104
8.8 Task state transitions.....	105
8.9 Task set management examples.....	106
8.9.1 Introduction.....	106
8.9.2 Head of queue tasks.....	107
8.9.3 Ordered tasks.....	108
8.9.4 ACA task.....	109
Annex A (informative) Identifiers and names for objects.....	110
A.1 Identifiers and names overview.....	110
A.2 Identifiers and names.....	110
A.3 SCSI transport protocol acronyms and bibliography.....	113
Annex B (informative) Terminology mapping.....	115
Bibliography.....	116

Tables

	Page
1 ISO/IEC and American numbering conventions examples	24
2 Single level logical unit number structure for logical unit numbers 255 and below	41
3 Single level logical unit number structure for logical unit numbers 16 383 and below	41
4 Eight byte logical unit number structure adjustments	42
5 Eight byte logical unit number structure	43
6 Format of addressing fields.....	43
7 ADDRESS METHOD field values.....	43
8 Peripheral device addressing.....	44
9 Flat space addressing	45
10 Logical unit addressing	45
11 Extended logical unit addressing	46
12 LENGTH field values and related sizes	46
13 Two byte extended logical unit addressing format	46
14 Four byte extended logical unit addressing format	46
15 Six byte extended logical unit addressing format.....	46
17 Logical unit extended addressing	47
18 Well-known logical unit extended address format.....	47
16 Eight byte extended logical unit addressing format	47
19 Logical unit not specified extended address format	48
20 Mapping nexus to SAM-2 identifiers	49
21 CONTROL byte	66
22 Status codes	67
23 Actions that affect task(s) received on this or other I_T nexuses.....	77
24 Task handling when ACA is not in effect	80
25 Aborting tasks when an ACA is not established	81
26 Blocking and aborting tasks when an ACA is established	82
27 Handling for new tasks received on a faulted I_T nexus during ACA	82
28 Handling for new tasks received on non-faulted I_T nexuses during ACA	83
29 Unit attention additional sense codes for events detected by SCSI target devices	88
30 Task Management Functions.....	92
31 Task State Nomenclature	101
32 Task attributes	103
33 Task attribute and state indications in examples	106
34 Dormant task blocking boundary requirements	108
A.1 Object size and support requirements	110
A.2 Object identifier size for each SCSI transport protocol.....	110
A.3 Object identifier format for each SCSI transport protocol	111
A.4 Object name size for each SCSI transport protocol	112
A.5 Object name format for each SCSI transport protocol.....	113
B.1 SAM-3 to SAM terminology mapping	115

Figures

	Page
0 SCSI document structure	8
1 Requirements precedence	10
2 Example hierarchy diagram	24
3 Example state diagram	27
4 Client-Server model	29
5 SCSI client - server model	30
6 SCSI I/O system and domain model	31
7 Overall SCSI domain model	32
8 SCSI domain model	32
9 SCSI initiator device model	35
10 SCSI target device model	36
11 SCSI target/initiator device with SCSI target/initiator ports model	37
12 SCSI target/initiator device without SCSI target/initiator ports model	37
13 Logical unit model	39
14 Eight byte logical unit number structure adjustments	42
15 SCSI device functional models	50
16 Multiple port target SCSI device structure model	51
17 Multiple port SCSI initiator device structure model	52
18 Multiple port target/initiator SCSI device structure model	53
19 SCSI target device configured in a single SCSI domain	54
20 SCSI target device configured in multiple SCSI domains	55
21 SCSI target device and SCSI initiator device configured in a single SCSI domain	56
22 Dependent logical unit model	57
23 Example of hierarchical system diagram	58
24 Protocol service reference model	59
25 SCSI transport protocol service model	60
26 Request-Response SAL transaction and related STPL services	61
27 SCSI transport protocol service model for data transfers	61
28 Device server data transfer transaction and related STPL services	62
29 SCSI transport protocol service model for Terminate Data Transfer	62
30 Device server Terminate Data Transfer transaction and related STPL services	63
31 Model for Data-In and Data-Out data transfers	72
32 Command processing events	78
33 Linked command processing events	79
34 Events and event notifications for SCSI target devices	87
35 Events and event notifications for SCSI initiator devices	88
36 Task management processing events	97
37 Example of Dormant state task behavior	102
38 Task states	105
39 Head of queue tasks and blocking boundaries (example 1)	107
40 Head of queue tasks and blocking boundaries (example 2)	107
41 Ordered tasks and blocking boundaries	108
42 ACA task example	109

INFORMATION TECHNOLOGY – SMALL COMPUTER SYSTEM INTERFACE (SCSI) –

Part 413: Architecture model-3 (SAM-3)

FOREWORD

- 1) ISO (International Organization for Standardization) and IEC (International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards. Their preparation is entrusted to technical committees; any ISO and IEC member body interested in the subject dealt with may participate in this preparatory work. International governmental and non-governmental organizations liaising with ISO and IEC also participate in this preparation.
- 2) In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.
- 3) The formal decisions or agreements of IEC and ISO on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC and ISO member bodies.
- 4) IEC, ISO and ISO/IEC Publications have the form of recommendations for international use and are accepted by IEC and ISO member bodies in that sense. While all reasonable efforts are made to ensure that the technical content of IEC, ISO and ISO/IEC Publications is accurate, IEC or ISO cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 5) In order to promote international uniformity, IEC and ISO member bodies undertake to apply IEC, ISO and ISO/IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any ISO/IEC Publication and the corresponding national or regional publication should be clearly indicated in the latter.
- 6) ISO and IEC provide no marking procedure to indicate their approval and cannot be rendered responsible for any equipment declared to be in conformity with an ISO/IEC Publication.
- 7) All users should ensure that they have the latest edition of this publication.
- 8) No liability shall attach to IEC or ISO or its directors, employees, servants or agents including individual experts and members of their technical committees and IEC or ISO member bodies for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication of, use of, or reliance upon, this ISO/IEC publication or any other IEC, ISO or ISO/IEC publications.
- 9) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 10) Attention is drawn to the possibility that some of the elements of this International Standard may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights

International Standard 14776-413 was prepared by subcommittee 25: Interconnection of information technology equipment, of ISO/IEC joint technical committee 1: Information technology.

This International Standard has been approved by vote of the member bodies, and the voting results may be obtained from the address given on the title page.

A list of all parts of the ISO/IEC 14776 series, under the general title *Information technology – Small computer system interface (SCSI)*, can be found on the IEC website.

This publication has been drafted in accordance with the ISO/IEC Directives, Part 2.

INTRODUCTION

SCSI standards family

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

Figure 0 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.

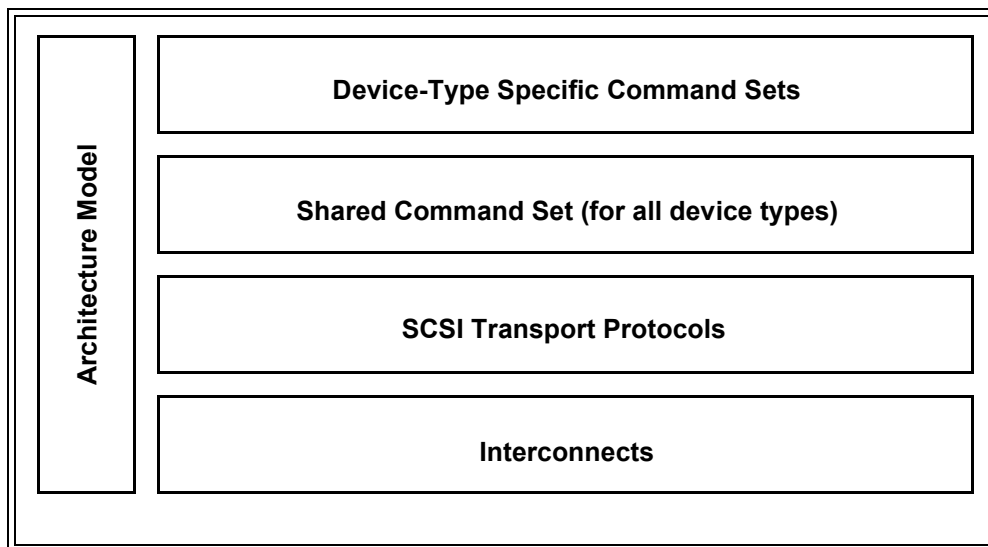


Figure 0 — SCSI document structure

The roadmap in figure 0 is intended to show the general applicability of the documents to one another. Figure 0 is not intended to imply a relationship such as a hierarchy, protocol stack or system architecture.

The functional areas identified in figure 0 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.

At the time this standard was generated, examples of the SCSI general structure included a number of Interconnects, SCSI Transport Protocols, Shared Command Sets, Device-Type Specific Command Sets and Architecture Models listed in the bibliography.

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.

The SCSI Architecture Model - 3 (SAM-3) 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.

Annex B identifies differences between the terminology used in this standard and previous versions of this standard.

Copyright International Organization for Standardization

INFORMATION TECHNOLOGY – SMALL COMPUTER SYSTEM INTERFACE (SCSI) –

Part 413: Architecture model-3 (SAM-3)

1 General

1.1 Scope

The set of SCSI (Small Computer System Interface) standards consists of this standard and the SCSI implementation standards described in the precedence requirements (see 1.2). 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 part of ISO/IEC 14776 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 edition:

- a) support for the SPI-5 SCSI transport protocol (except for informational listings in Annex A);
- b) contingent allegiance;
- c) the TARGET RESET task management function and
- d) untagged tasks.

1.2 Precedence requirements

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.

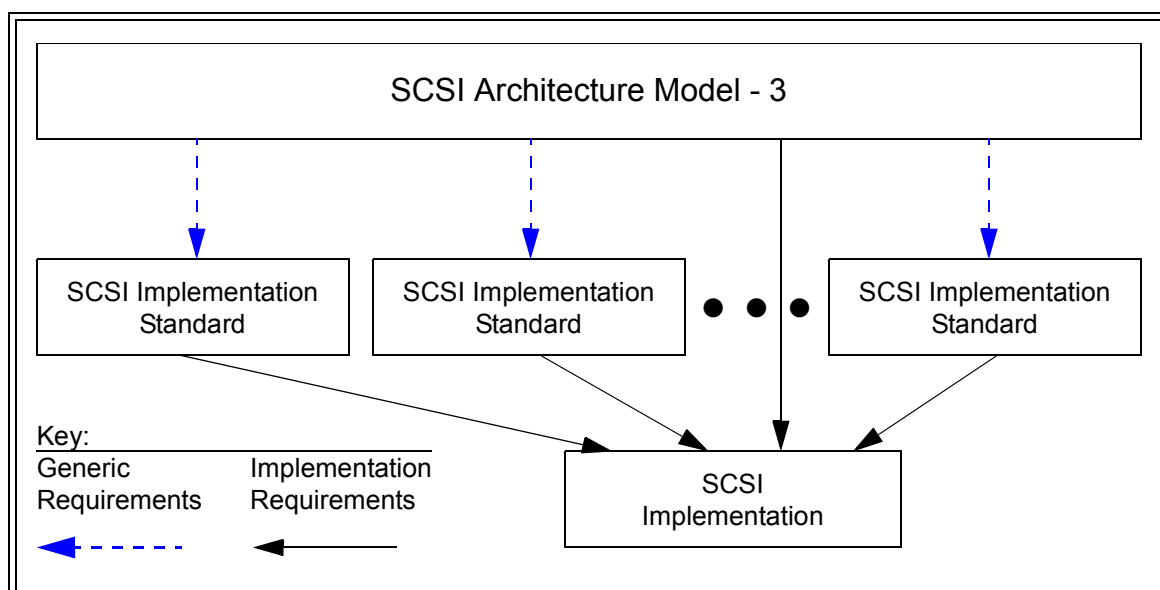


Figure 1 — Requirements precedence

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, the requirements of this standard shall apply.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 14776-322, *Information technology – Small computer system interface (SCSI) – Part 322: Block commands-2 (SBC-2)* [T10/1417-D]

ISO/IEC 14776-453, *Information technology – Small computer system interface (SCSI) – Part 453: Primary commands-3 (SPC-3) (under consideration)* [T10/1416-D]

,

3 Definitions, symbols, abbreviations, and conventions

3.1 Definitions

3.1.1

ACA command: command performed by a task with the ACA attribute (see 3.1.5, 4.11, and 8.6.5)

3.1.2

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

3.1.3

application client: an object that is the source of commands and task management function requests

3.1.4

argument: a datum provided as input to or output from a procedure call (see 3.1.76)

3.1.5

auto contingent allegiance (ACA): 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 5.9.2.

3.1.6

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

3.1.7

basic task management model: task management model in which only one task attribute (see 8.6) is supported and the task management features that the application client may select are limited. See 8.3.3.

3.1.8

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

3.1.9

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

3.1.10

byte: an 8-bit construct

3.1.11

client-server: 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

3.1.12

client: an entity that requests a service from a server. This standard defines the application client

3.1.13

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 3.1.37), that is in a procedure call input argument (see 3.6.2), in a procedure call output argument or in a procedure call result

3.1.14

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

3.1.15

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 bytes and 260 bytes. See 5.2 and SPC-3

3.1.16

command standard: 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).

3.1.17

completed command: command that has ended by returning a status and service response of TASK COMPLETE or LINKED COMMAND COMPLETE

3.1.18

completed task: task that has ended by returning a status and service response of TASK COMPLETE. The actual events comprising the TASK COMPLETE response are SCSI transport protocol specific

3.1.19

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

3.1.20

confirmed SCSI transport protocol service: a service available at the SCSI transport protocol service interface that includes a confirmation of completion. See 4.15.

3.1.21

current task: 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

3.1.22

deferred error: error generated by a background operation (see 3.1.6)

3.1.23

dependent logical unit: logical unit that is addressed via some other logical unit(s) in a hierarchical logical unit structure (see 3.1.41), also a logical unit that is at a higher numbered level in the hierarchy than the referenced logical unit (see 4.14)

3.1.24

device identifier: term used by previous versions of this standard (see B)

3.1.25

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

3.1.26

device server: an object within a logical unit that processes SCSI tasks according to the requirements for task management described in clause 8

3.1.27

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

3.1.28

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

3.1.29

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 the service delivery subsystem

3.1.30

dormant task state: when in this state a task is prevented from entering the enabled task state (see 3.1.31) due to the presence of certain other tasks in the task set

3.1.31

enabled task state: when in this state a task may complete at any time or is waiting to receive the next command in a series of linked commands

3.1.32

extended logical unit addressing: the logical unit addressing method is used by special function logical units (e.g., well known logical units). See 4.9.9.

3.1.33

faulted I_T nexus: 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

3.1.34

faulted task set: 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

3.1.35

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

3.1.36

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

3.1.37

field: a group of one or more contiguous bits, part of a larger structure (e.g., a CDB (see 3.1.15) or sense data (see 3.1.108))

3.1.38

full task management model: the task management model in which the SIMPLE task attribute (see 8.6.1) is supported and all task management features may be selected by the application client, see 8.3.2

3.1.39

function complete: a logical unit response indicating that a task management function has finished. The events comprising this response are SCSI transport protocol specific

3.1.40

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

3.1.41

hierarchical logical unit: an inverted tree structure for forming and parsing logical unit numbers (see 3.1.64) containing up to four addressable levels (see 4.14)

3.1.42

I_T nexus: nexus between a SCSI initiator port and a SCSI target port. See 4.12.

3.1.43

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

3.1.44

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

3.1.45

I_T_L nexus: nexus between a SCSI initiator port, a SCSI target port, and a logical unit (see 4.12)

3.1.46

I_T_L_Q nexus: nexus between a SCSI initiator port, a SCSI target port, a logical unit and a task. See 4.12

3.1.47

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

3.1.48

I_T_L_x nexus: either an I_T_L nexus or an I_T_L_Q nexus. See 4.12.

3.1.49

I/O operation: operation defined by an unlinked command, a series of linked commands or a task management function

3.1.50

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

3.1.51

initiator: term used in previous versions of this standard (see B)

3.1.52

initiator device name: SCSI device name of a SCSI initiator device. See 4.7.2.

3.1.53

initiator identifier: term used in previous versions of this standard (see B)

3.1.54

initiator port identifier: a value by which a SCSI initiator port is referenced within a domain. See 4.7.2.

3.1.55

initiator port name: SCSI port name (see 3.1.97) of a SCSI initiator port or of a SCSI target/initiator port when operating as a SCSI initiator port. See 4.7.2.

3.1.56

in transit: information that has been delivered to the service delivery subsystem for transmission, but that has not been received yet

3.1.57

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

3.1.58

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

3.1.59

linked command: one in a series of commands processed by a single task that collectively make up a discrete I/O operation. In such a series, each command is represented by the same I_T_L_Q nexus and all, except the last, have the LINK bit in the CDB CONTROL byte set to one

3.1.60

logical unit: 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 4.8.

3.1.61

logical unit reset: 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

3.1.62

logical unit reset event: an event that results in a logical unit reset condition as described in 6.3.3

3.1.63

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

3.1.64

logical unit number (LUN): a 64-bit or 16-bit identifier for a logical unit. See 4.9.

3.1.65

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

3.1.66

nexus: relationship between two SCSI devices and the SCSI initiator port and SCSI target port objects within those SCSI devices. See 4.12.

3.1.67

non-faulted I_T nexus: I_T nexus that is not a faulted I_T nexus (see 3.1.33)

3.1.68

object: container that encapsulates data types, services or other objects that are related in some way

3.1.69

peer entities: entities within the same layer (see 3.1.58)

3.1.70

pending command: from the point of view of the application client, the description of a 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

3.1.71

port: synonymous with SCSI port (see 3.1.95 and Annex B)

3.1.72

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

3.1.73

power on: 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

3.1.74

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

3.1.75

procedure: operation that is invoked through an external calling interface

3.1.76

procedure call: the model used by this standard for the interfaces involving both the SAL (see 3.1.86) and STPL (see 3.1.102), having the appearance of a programming language function call

3.1.77

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

3.1.78

queue: arrangement of tasks within a task set (see 3.1.130), usually according to the temporal order in which they were created

3.1.79

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

3.1.80

reference model: a standard model used to specify system requirements in an implementation independent manner

3.1.81

relative port identifier: identifier for a SCSI port that is unique within a SCSI device. See 4.7.6.

3.1.82

request: a transaction invoking a service

3.1.83

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

3.1.84

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

3.1.85

response: a transaction conveying the result of a request

3.1.86

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

3.1.87

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

3.1.88

SCSI device identifier: synonymous with SCSI port identifier (see 3.1.96)

3.1.89

SCSI device name: name (see 3.1.65) 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 4.7.8). The SCSI device name may be made available to other SCSI devices or SCSI ports in SCSI transport protocol specific ways

3.1.90

SCSI event: condition defined in 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.

3.1.91

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

3.1.92

SCSI identifier: term used in previous versions of this standard (see B)

3.1.93

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 this term is used, it refers to SCSI initiator devices or SCSI target/initiator devices that are using the SCSI target/initiator port as a SCSI initiator port

3.1.94

SCSI initiator port: a SCSI initiator device object that acts as the connection between application clients and the 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 or a SCSI target/initiator port operating as a SCSI initiator port

3.1.95

SCSI port: SCSI device resident object that connects the application client, device server or task manager to the 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 3.1.94), a SCSI target port (see 3.1.99) or a SCSI target/initiator port (see 3.1.101)

3.1.96

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 3.1.54) or a target port identifier (see 3.1.119)

3.1.97

SCSI port name: name (see 3.1.65) of a SCSI port that is world-wide unique within the SCSI transport protocol of the SCSI domain of that SCSI port (see 4.7.9). The name may be made available to other SCSI devices or SCSI ports in that SCSI domain in SCSI transport protocol specific ways

3.1.98

SCSI target device: 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 this term is used, it refers to SCSI target devices or SCSI target/initiator devices that are using the SCSI target/initiator port as a SCSI target port

3.1.99

SCSI target port: SCSI target device object that contains a task router and acts as the connection between device servers and task managers and the service delivery subsystem through which indications and responses are routed. When this term is used, it refers to a SCSI target port or a SCSI target/initiator port operating as a SCSI target port

3.1.100

SCSI target/initiator device: SCSI device that has all the characteristics of a SCSI target device and a SCSI initiator device

3.1.101

SCSI target/initiator port: SCSI device resident object that has all the characteristics of a SCSI target port and a SCSI initiator 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 3.1.47) as a CHECK CONDITION status (see 5.9.6). 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: SENSE KEY field in the sense data (see 3.1.108 and SPC-3)

3.1.110

server: 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 the 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 is 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: a term used in previous versions of this standard (see B)

3.1.117

target device name: SCSI device name (see 3.1.89) of a SCSI target device. See 4.7.3.

3.1.118

target identifier: a term used in previous versions of this standard (see B)

3.1.119

target port identifier: a value by which a SCSI target port is referenced within a domain. See 4.7.3.

3.1.120

target port name: SCSI port name of a SCSI target port or of a SCSI target/initiator port when operating as a SCSI target port. See 4.7.3.

3.1.121

target/initiator device name: SCSI device name (see 3.1.89) of a SCSI target/initiator device. See 4.7.4.

3.1.122

task: an object within the logical unit representing the work associated with a command or a group of linked commands. See 4.11.

3.1.123

task priority: the relative scheduling importance of a task having a SIMPLE task attribute in relation to other tasks already in the task set. See 8.7.

3.1.124

task tag: an object containing up to 64 bits that uniquely identifies each task for a given I_T_L nexus (see 3.1.45) in a task set (see 3.1.130). See 4.11.

3.1.125

task management function: a task manager service capable of being requested by an application client to affect the processing of one or more tasks

3.1.126

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

3.1.127

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

3.1.128

task manager: server within a logical unit that controls the sequencing of one or more tasks and processes task management functions

3.1.129

task router: an object in a SCSI target port that routes commands and task management functions between the service delivery subsystem (see 3.1.113) and the appropriate logical unit's task manager (see 3.1.128)

3.1.130

task set: group of tasks within a logical unit, whose interaction is dependent on the task management (e.g., queuing) and ACA requirements. See 4.8.

3.1.131

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

3.1.132

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

3.1.133

unconfirmed SCSI transport protocol service: a service available at the SCSI transport protocol service interface that does not result in a completion confirmation. See 4.15.

3.1.134

unlinked command: command having the LINK bit set to zero in the CDB CONTROL byte that is not the last command in a series of linked commands (see 3.1.59)

3.1.135

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 4.10.

3.1.136

well known logical unit number (W-LUN): the logical unit number that identifies a well known logical unit. See 4.9.10.

3.2 Acronyms

ACA	Auto Contingent Allegiance (see 5.9.2)
CDB	Command Descriptor Block (see 5.2)
CRN	Command Reference Number (see 5.1)

EUI	Extended Unique Identifier (see http://standards.ieee.org/regauth/oui/tutorials/UseOfEUI.html)
FCP-2	Fibre Channel Protocol for SCSI, Version 2 (see Bibliography)
I/O	Input / Output
iSCSI	internet SCSI (see RFC 3720)
ISID	Initiator Session Identifier (see RFC 3720)
ISO	Organization for International Standards
LUN	Logical Unit Number
LSB	Least significant bit
MMC-2	SCSI Multi-Media Commands -2 (see Bibliography)
MSB	Most significant bit
NACA	Normal ACA (see 5.2)
n/a	Not Applicable
RAID	Redundant Array of Independent Disks
SAL	SCSI application layer (see 3.1.86)
SAS	Serial Attached SCSI (see Bibliography)
SBC-2	SCSI-3 Block Commands -2 (see clause 2)
SBP-3	Serial Bus Protocol -3 (see Bibliography)
SCSI	The architecture defined by the family of standards described in the Bibliography
SPI-5	SCSI Parallel Interface -5 (see Bibliography)
SPC-3	SCSI Primary Commands -3 (see clause 2)
STPL	SCSI transport protocol layer (see Bibliography)
SRP	SCSI RDMA Protocol (see Bibliography)
SSC	SCSI-3 Stream Commands (see Bibliography)
SSP	Serial SCSI Protocol (see SAS)
TPGT	Target Portal Group Tag (see RFC 3720)
TST	Task Set Type (see SPC-3)
UCS	Universal Character Set (see ISO/IEC 10646-1)
UTF	UCS Transformation Formats (see RFC 2279)
VPD	Vital Product Data (see SPC-3)
W-LUN	Well known logical unit number

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 in 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 Bibliography).

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 the sake of 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 22 (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; then 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 comprised 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 comprised 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 comprised 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 and IEC 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 using the ISO/IEC and American conventions.

Table 1 — ISO/IEC and American numbering conventions examples

ISO/IEC	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 Hierarchy diagram conventions

Hierarchy diagrams show how objects are related to each other. The hierarchy diagram of figure 2, for example, shows the relationships among the objects comprising an object called Book. For this example, a book object is defined as containing a table of contents object, an optional preface object, one or more chapter objects and an optional index object. Further contents definitions are provided for the preface and chapter objects. A preface object contains zero or more figure objects, one outline object and an introductory text object. A chapter object contains one or more section objects and zero or more figure objects.

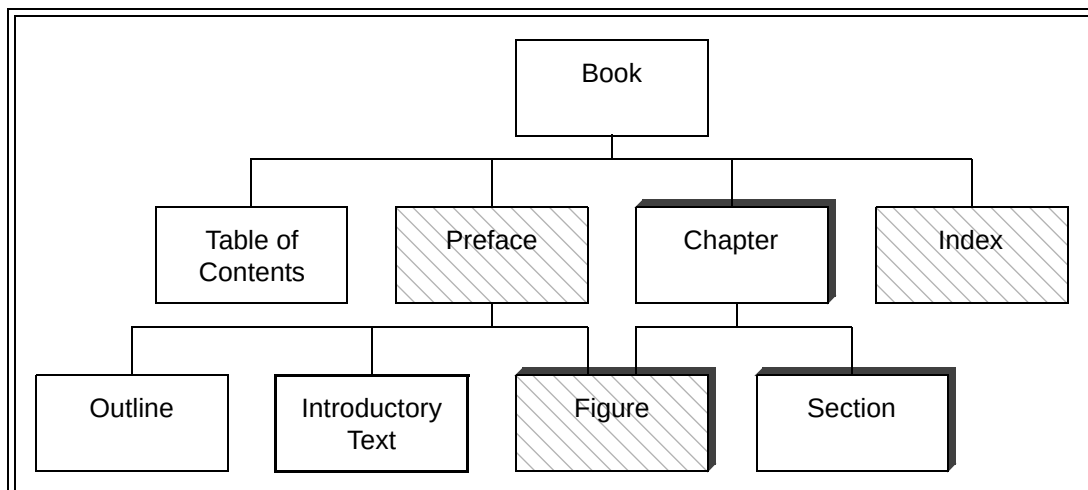


Figure 2 — Example hierarchy diagram

In the corresponding hierarchy diagram, labelled boxes denote the above objects. The composition and relation of one object to others is shown by the connecting lines. In this case, the connecting lines indicate the relationship between the book object and its constituent table of contents, preface, chapter and index objects. Similarly, connecting lines show that a chapter object contains section and figure objects. Note that the figure object also may be a component of the preface object.

In the hierarchy diagram, objects that are required to have one and only one instance are shown as simple boxes, as is the case for the book and table of contents objects. The hierarchy diagram shows multiple instances of an object by the presence of a shadow, as is the case for the chapter, figure and section objects. Objects that are optional are indicated by light diagonal lines, as is the case for the preface, figure and index objects. An object that may not have any instances, have only one instance, or have multiple instances is shown with both diagonal lines and a shadow, as is the case for the figure object. The instance indications shown in a hierarchy diagram are approximate; detailed requirements appear in the accompanying text.

Figure 1. Hierarchy diagram

3.6.2 Notation for procedure calls

In this standard, the model for functional interfaces between entities is a procedure call (see 3.1.76). 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.

3.6.3 Notation for state diagrams

All state diagrams use the notation shown in figure 3.

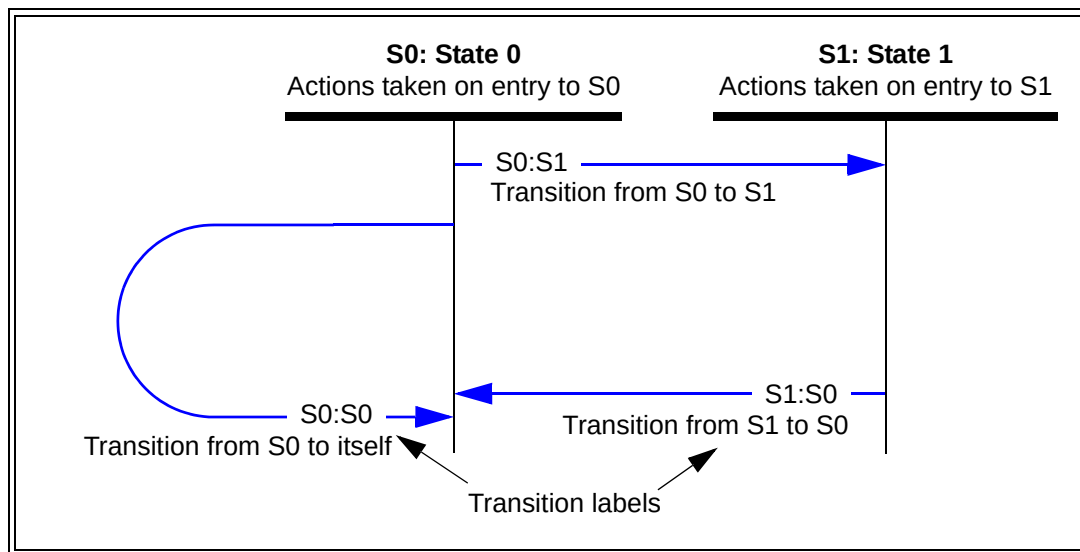


Figure 3 — Example state diagram

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 3 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:

- time elapses only within discrete states;
- state transitions are logically instantaneous and
- 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 objects (see 3.1.68), protocol layers and service interfaces between objects. As used in this standard, objects are abstractions, encapsulating a set of related functions, data types and other objects. Certain objects 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 objects exhibit well-defined and observable behaviors, but they do not exist as separate physical elements. An object may be a single numeric parameter (e.g., a task tag) or a complex entity that performs a set of operations or services on behalf of another object.

Service interfaces are defined between distributed objects 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 objects. 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 7.8. The model describes required behavior in terms of layers, objects within layers and SCSI transport protocol service transactions between layers.

4.2 The SCSI distributed service model

Service interfaces between distributed objects are represented by the client-server model shown in figure 4. Dashed horizontal lines with arrowheads denote a single request - response transaction as it appears to the client and server. The solid lines with arrowheads indicate the actual transaction path through the service delivery subsystem. In such a model, each client or server is a single thread of processing that runs concurrently with all other clients or servers.

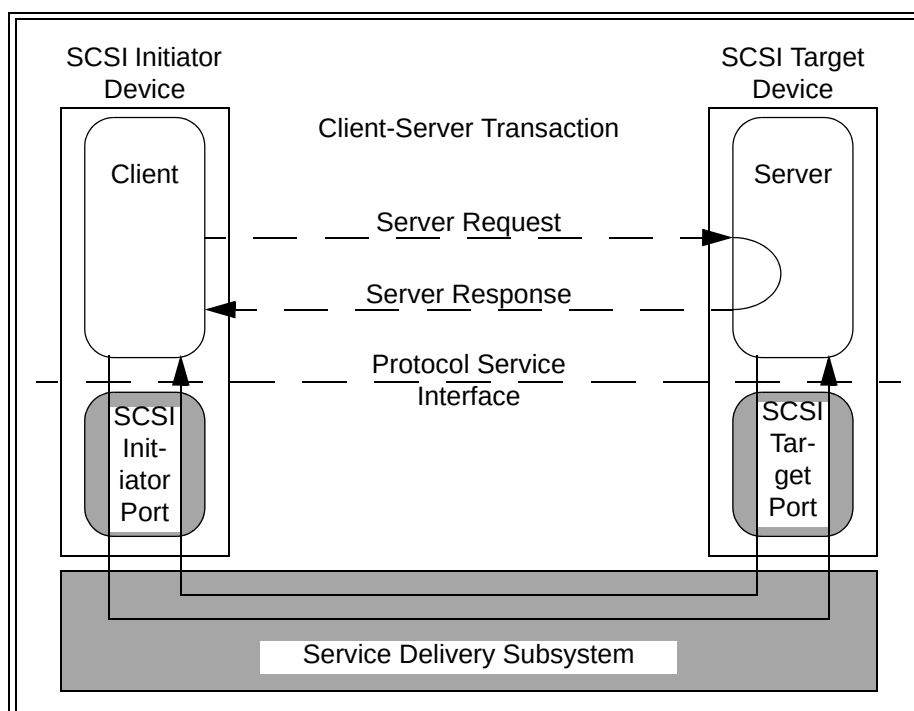


Figure 4 — Client-Server model

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. The function of the service delivery subsystem is to transport error-free copies of the requests and responses between sender and receiver. 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 5.6).

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

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

4.3 The SCSI client - server model

As shown in figure 5, each SCSI target device contains one or more logical units and provides services performed by device servers and task management functions performed by task managers. A logical unit is an object that implements one of the device functional models described in the SCSI command standards and processes commands (e.g., reading from or writing to the media). Each pending command or series of linked commands defines a unit of work to be performed by the logical unit. Each unit of work is represented within the SCSI target device by a task that may be externally referenced and controlled through requests issued to the task manager.

© 2007 International Organization for Standardization

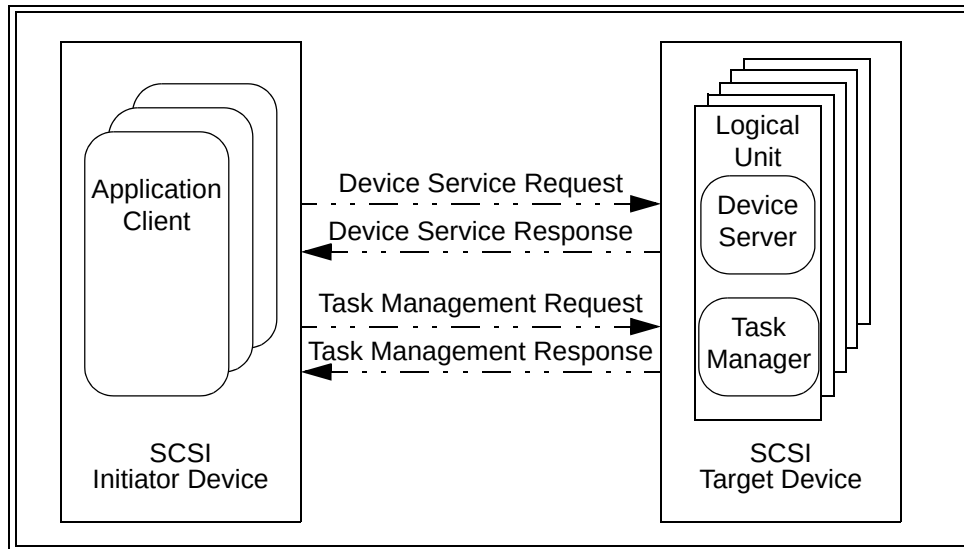


Figure 5 — SCSI client - server model

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

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

A task is an object within the logical unit representing the work associated with a command or series of linked commands. A new command or the first in a series of linked commands 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. An example of the processing for a single command is given in 5.8.1. An example of linked command processing is given in 5.8.2.

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 7.9.

4.4 The SCSI structural model

The SCSI structural model represents a view of the elements comprising a SCSI I/O system as seen by the application clients interacting with the system. As shown in figure 6, the fundamental object is the SCSI domain that represents an I/O system. A 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.

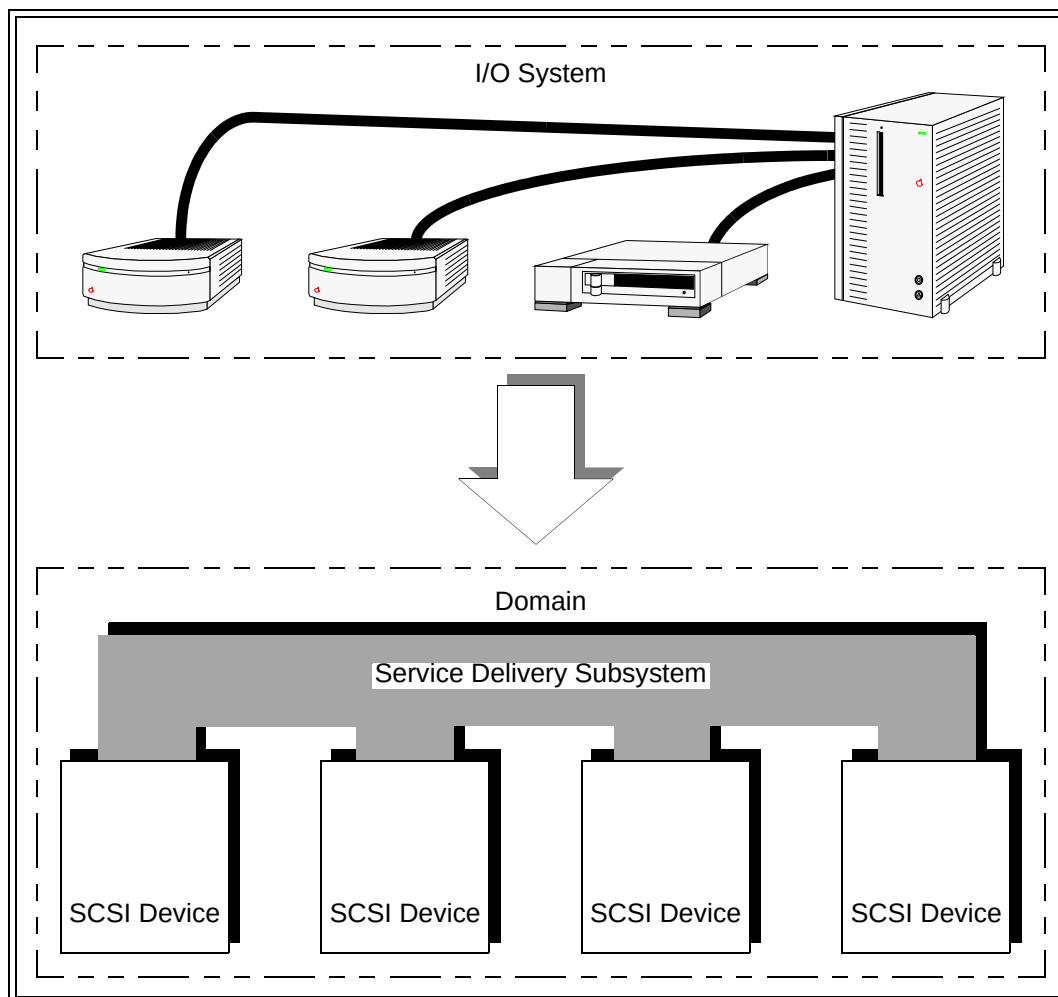


Figure 6 — SCSI I/O system and domain model

Copyright International Organization for Standardization

Figure 7 shows the main functional components of the SCSI domain. This standard defines these components in greater detail.

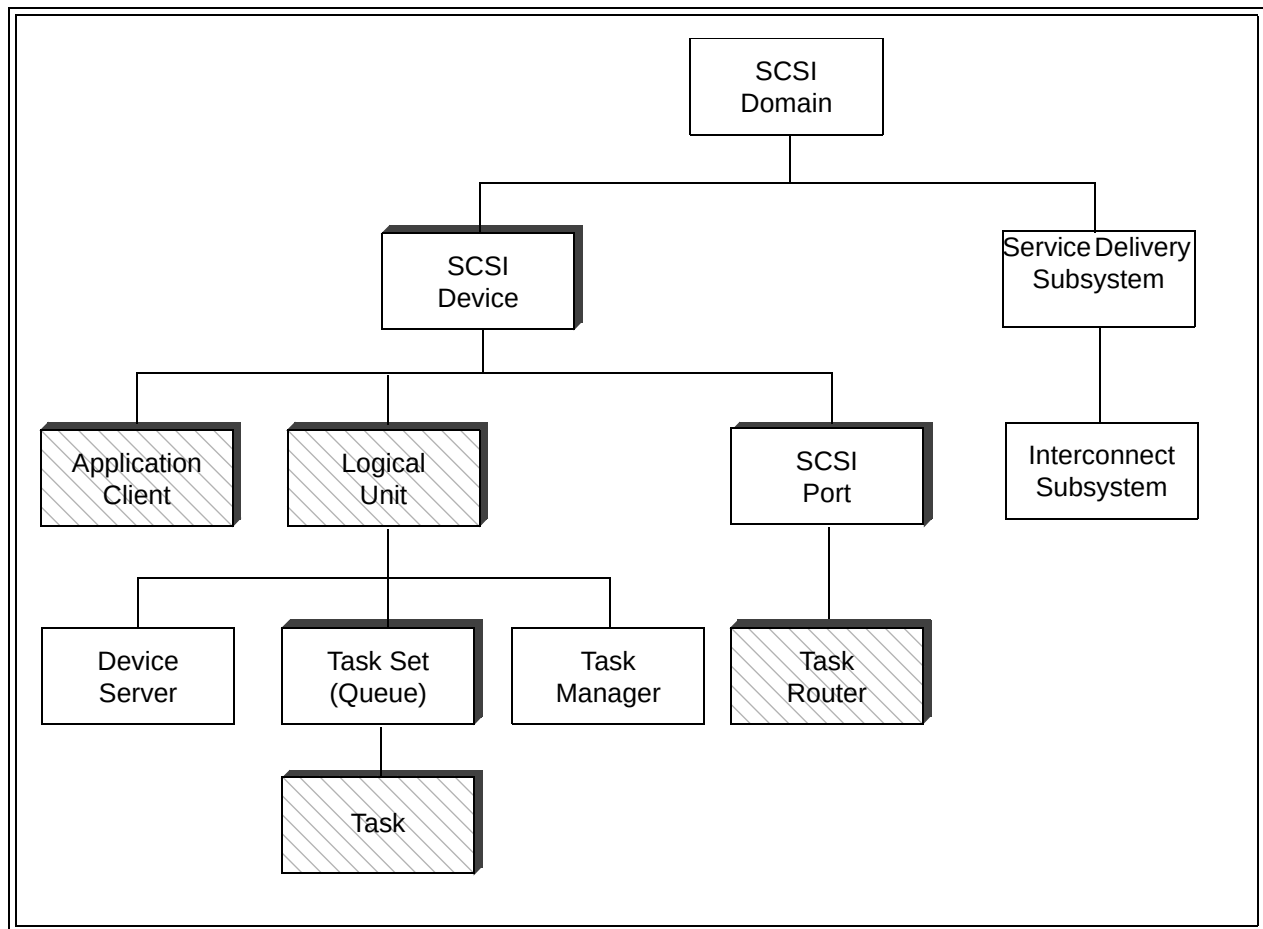


Figure 7 — Overall SCSI domain model

4.5 SCSI domain

A SCSI domain is composed of at least one SCSI device, at least one SCSI target port and at least one SCSI initiator port interconnected by a service delivery subsystem (see figure 8).

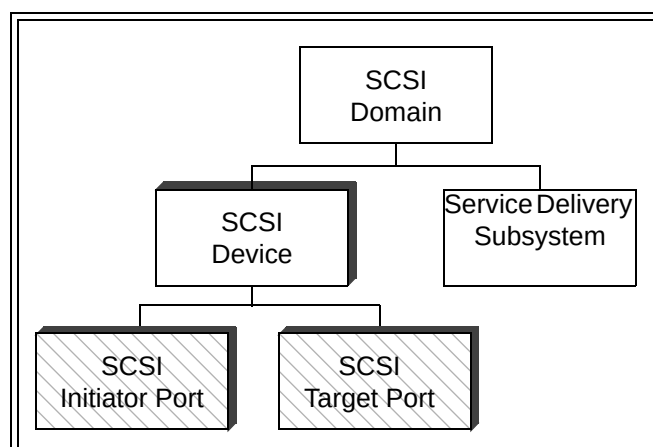


Figure 8 — SCSI domain model

A SCSI device (see 4.7) is an object that originates or processes commands and task management functions.

When a SCSI device originates a command or task management function it is called a SCSI initiator device. The commands and task management functions are transmitted through SCSI initiator ports or SCSI target/initiator ports.

A SCSI device containing one or more logical units that process commands is called a SCSI target device. It receives commands and task management functions through SCSI target ports or SCSI target/initiator ports.

The service delivery subsystem connects all the SCSI ports in the SCSI domain, providing a mechanism through which application clients communicate with device servers and task managers (see 4.6).

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

4.6 The service delivery subsystem

4.6.1 The service delivery subsystem object

The service delivery subsystem connects SCSI ports (see 3.1.95) and is composed of one or more interconnects that appear to a client or server as a single path for the transfer of requests and responses between SCSI devices.

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

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

4.6.2 Synchronizing client and server states

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

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

This standard assumes that state synchronization, if required by a SCSI transport protocol standard, is enforced by the service delivery subsystem transparently to the server (i.e., whenever the server invokes a SCSI transport protocol service to return a response as described in 7.8 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).

4.6.3 Request/Response ordering

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

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

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

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

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

4.7 SCSI devices

4.7.1 General

A SCSI device is a SCSI target device, a SCSI initiator device or a SCSI target/initiator device.

A SCSI initiator device contains at least one SCSI initiator port and is capable of originating commands and task management requests (see 4.7.2). A SCSI target device contains at least one SCSI target port, at least one logical unit and is capable of processing commands and task management requests (see 4.7.3). A SCSI target/initiator device contains at least one SCSI target/initiator port, at least one logical unit, and it is capable of originating and processing commands and task management requests (see 4.7.4).

To be functional, a SCSI domain is required to contain a SCSI target port or a SCSI target/initiator port operating as a SCSI target port and a SCSI initiator port or SCSI target/initiator port operating as a SCSI initiator port.

4.7.2 SCSI initiator device

A SCSI initiator device (see figure 9) contains:

- a) zero or more initiator device names;
- b) one or more SCSI initiator ports, each containing an
 - A) initiator port identifier and
 - B) optional initiator port name;
- c) one or more application clients and

d) zero or more application client tasks.

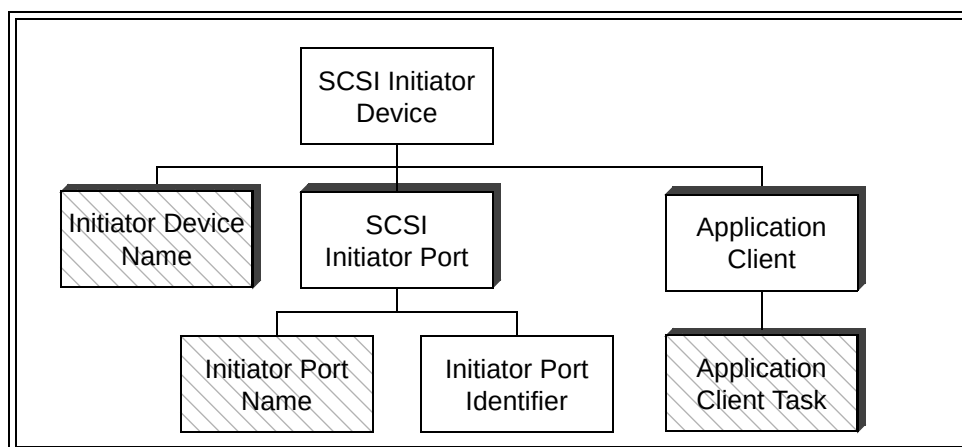


Figure 9 — SCSI initiator device model

The initiator port identifier is a value that is the SCSI port identifier (see 4.7.5) for a SCSI initiator port.

An initiator device name is a name (see 3.1.65) that is a SCSI device name (see 4.7.8) for a SCSI initiator device. For each supported SCSI transport protocol, a SCSI initiator device shall have no more than one (i.e., zero or one) SCSI initiator device name that is not in the SCSI name string format (see SPC-3). A SCSI initiator device shall have no more than one (i.e., zero or one) SCSI initiator device name in the SCSI name string format regardless of the number of SCSI transport protocols supported by the SCSI initiator device. If a SCSI initiator device has a SCSI device name in the SCSI name string format then the SCSI initiator device should have only one SCSI initiator device name. A SCSI transport protocol standard may place additional requirements on initiator device names.

The initiator port name is a name (see 3.1.65) that is the SCSI port name (see 4.7.9) for the SCSI initiator port. A SCSI transport protocol standard may place additional requirements on initiator port names.

Application clients are the sources of commands and task management functions.

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

4.7.3 SCSI target device

A SCSI target device (see figure 10) contains:

- a) zero or more target device names;
- b) one or more SCSI target ports, each containing
 - A) a task router;
 - B) a target port identifier,
 - C) an optional relative port identifier and
 - D) an optional target port name;
 and