Working Draft

T10 Project 1561-D

Revision 2
11 September 2002

Information technology - SCSI Architecture Model - 3 (SAM-3)

T10 Technical Editor:

Ralph O. Weber
ENDL Texas
18484 Preston Road
Suite 102 PMB 178
Dallas, TX 75252
USA

Telephone: 214-912-1373
Facsimile: 972-596-2775
Email: ROWeber@ACM.org

Reference number
ISO/IEC ISO/IEC 14776-413-200x
ANSI INCITS.*:200x**

Points of Contact:

T10 Chair
John B. Lohmeyer
LSI Logic
4420 Arrows West Drive
Colorado Springs, CO 80907-3444
Tel: (719) 533-7560
Fax: (719) 533-7183
Email: lohmeyer@t10.org

T10 Vice-Chair
George O. Penokie
IBM
3605 Highway 52 N
MS: 2C6
Rochester, MN 55901
Tel: (507) 253-5208
Fax: (507) 253-2880
Email: gop@us.ibm.com

INCITS Secretariat
INCITS Secretariat
1250 Eye Street, NW Suite 200
Washington, DC 20005
Telephone: 202-737-8888
Facsimile: 202-638-4922
Email: INCITS@itic.org

T10 Web Site www.t10.org

T10 Reflector To subscribe send e-mail to majordomo@T10.org with 'subscribe' in message body
To unsubscribe send e-mail to majordomo@T10.org with 'unsubscribe' in message body
Internet address for distribution via T10 reflector: T10@T10.org

Document Distribution

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

or

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

ANSI (r)
INCITS.***:200x

Draft

American National Standards for Information Systems -

SCSI Architecture Model - 3 (SAM-3)

Secretariat
InterNational Committee for Information Technology Standards

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.

Draft

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer. Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered and that effort be made toward their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he or she has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not confirming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give interpretation on any American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

CAUTION: The developers of this standard have requested that holders of patents that may be required for the implementation of the standard, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this standard.

As of the date of publication of this standard and following calls for the identification of patents that may be required for the implementation of the standard, no such claims have been made. No further patent search is conducted by the developer or the publisher in respect to any standard it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this standard.

Published by
American National Standards Institute
11 West 42nd Street, New York, NY 10036

Printed in the United States of America

Contents

Tables

Figures

Revision Information

1 Approved Documents Included

The following T10 approved proposals have been incorporated SAM-3 up to and including this revision:

02-155r6 Response to T10 Letter Ballot comments on SAM-2
02-227r1 Rewrite of SAM-2 example for dependent logical units to remove SCC-isms
02-241r2 Making Application Clients and Device Servers True Peers in SAM-2
02-244r0 SAM to SAM-2 Changed Terms Annex
02-245r0 SAM-3 Identifiers and names for SAS SSP
02-273 Minutes of T10 Meeting #50 — Change AER protocol requirement from 'shall' to 'may'
02-278r0 SAM-3 Query Task
02-348r0 SAM-3 Names and identifiers for iSCSI revision 16

The following proposal has been approved by T10 approved but is not included in revision 2 because revision 2 focuses solely on the changes approved to resolve the SAM-2 letter ballot comments:

02-232r2 Clearing effects of I_T nexus loss

This proposal will be incorporated in SAM-2 revision 3.

2 Revision History

2.1 Revision 0 (12 July 2002)

SAM-3 revision 0 is exactly SAM-2 revision 23 (i.e., the revision of SAM-2 on which T10 conducted a Letter Ballot in April 2002), except that all change bars have been removed. This prepares SAM-3 to assist T10 in following SAM changes resulting from the Letter Ballot, because SAM-3 revision 1 can include those changes with change bars, an aid that will not be placed in the Public Review revision of SAM-2.

2.2 Revision 1 (22 July 2002)

The following T10 approved proposals were incorporated in SAM-3 in this revision:

02-245r0 SAM-3 Identifiers and names for SAS SSP
02-273 Minutes of T10 Meeting #50 — Change AER protocol requirement from 'shall' to 'may'
02-278r0 SAM-3 Query Task

No other changes were made in revision 1.

2.3 Revision 2 (11 September 2002)

The following proposals were incorporated SAM-3 in this revision because the CAP working group approved them as part of the SAM-2 letter ballot resolution:

02-155r6 Response to T10 Letter Ballot comments on SAM-2
02-227r1 Rewrite of SAM-2 example for dependent logical units to remove SCC-isms
02-241r2 Making Application Clients and Device Servers True Peers in SAM-2
02-244r0 SAM to SAM-2 Changed Terms Annex
02-348r0 SAM-3 Names and identifiers for iSCSI revision 16

Revision 2 of SAM-3 incorporates all the CAP approved changes in response to the SAM-2 letter ballot. Because of coordination issues, these changes have not been approved by T10 as of the time SAM-3 revision 2 is being prepared. If changes are not approved by T10, then they will be removed in revision 3.

3 Plans for Future Revisions

This is a list of the work the technical editor considers required in future revisions of SAM-x.

3.1 Minor Changes

The terms “call”, “procedure”, and any related terms should have glossary definitions that clearly identify them as architectural abstractions. All of these concepts are wording conveniences used as shorthand by the architecture and model to express more complex concepts or concepts for which numerous implementations are possible. The technical editor also should search on the terms “call” and “procedure” to locate any uses and edit text at each usage point to clearly identify “call” and “procedure” as architectural model abstractions and not as indications of implementation requirements. In a similar vein, “protocol” is an architectural abstraction, however this may be better understood as an abstraction in the community of SCSI designers.

The term “service” appears to be an architectural abstraction too, it is defined totally on architectural abstractions (“calls” and “objects”). However, careful study is required to determine if “service” has some non-abstract, concrete meaning. If it does, the glossary definition should be changed.

Is it necessary to have definitions for “implementation option”, “logical unit option”, and “protocol option”? Surely, an option is an option is an option and the context in which the word “option” appears is sufficient to identify whose option is being discussed.

The technical editor wishes to remove the definition of “ended command”. Strictly speaking, the term “ended command” is not used anywhere in the working draft, thus allowing removal of the definition as a strictly editorial change. However, some consideration of the change appears prudent. The word “ended” is used frequently in the working draft, all uses appear to have the standard English meaning, but this thesis needs additional verification. Also, there is an “ended (task state)” that lacks a glossary definition and perhaps should have one.

The glossary definition of “layer” in uninformative, owing in part to the vague usage of “rank”. A clearer, more specific definition is needed.

Is the term “protocol service request” really so general as to require its being defined in terms of “call” (an architectural abstraction)? Also, the technical editor believes that “protocol service response” should be defined as “A reply to the upper level protocol …”, not “A reply from the upper level protocol …” Finally, would it be possible to cast both definitions in terms of specific entities from the SCSI roadmap, instead of “lower level protocol” and “upper level protocol” (see 3.2)?

Although it is used in the Foreword, Scope, and one figure title, the term “reference model” is little more than obfuscated wording for “model”. Could it be replaced?

Is “subsystem” really used as it is defined in the glossary? Many other SCSI standards use subsystem differently.

It seems that task management function names sometimes appear in all capitals and bold, not capitalized and bold as is stated in 3.4.

Why is the following sentence from the end of the first paragraph in 4.2 so important:

> “In such a model, each client or server is a single thread of execution which runs concurrently with all other clients or servers.”

Are the requirements on protocols really contained only in 5.4 and 6.10, as is stated in 4.1?

Change all usage of “remote procedure call” to “procedure call”, since “remote procedure call” is not defined.

Clause 5.3 is a mess. The data delivery services are given individual 5.3.x clauses but the command protocol services are not. 5.3.1 fails to identify the party responsible for establishing the parameters for the transfer of a buffer segment. The rule prohibiting input and output transfers by a single command is buried in a paragraph that starts with a discussion of buffer segmentation. The technical editor was very tempted to rewrite the whole clause during the revision 4 conversion, but wrote this reminder to himself instead.

5.6.4.1 seems to imply that other methods for controlling AER besides the Control mode page are acceptable. Is this really the intent of T10?

3.2 Substantial Changes

Is it really necessary for SAM-2 to place requirements on the contents of other standards? Would the SCSI documents set be just as well served if SAM-2 acted as a guide to what readers might expect to find in other SCSI standards? With these thoughts in mind, a few (but not necessarily all) specific instances of needed changes are noted:

a) The Foreword and Introduction clauses need to be modified to remove the work “requirements”; at the time of this writing, “capabilities” is the preferred replacement;
b) Most of 1.1 probably would be obsolete; and
c) A careful audit of the requirements statements will be needed to adjust those placing requirements on other standards.

The technical editor is considering a careful review of the working draft, with an eye toward overly abstract model abstractions. Examples are:

a) Overly general layering terms and discussions; and
b) Discussion of a new application client for each new request or task management function.

The layering seems overly general and thus confusing. SCSI has two (or at most three) layers. The question of two or three layers depends on whether the service delivery port is a layer. The two “main” layers are the command and control layer (application client, device server, and task manager) and the service delivery subsystem. The description appears amenable to substantial simplifications. LLP and ULP could disappear. Generalized interfaces could be replaced with a small number of specific interfaces. Does T10 see value in this kind of simplification?

The terms “SCSI application layer” and “SCSI protocol layer” appear to be redundant. Certainly, “SCSI application layer” is little more than a generalization of “application client”. Perhaps, “SCSI application layer” and “SCSI protocol layer” can be removed. As if this confusion were not enough, the definition of “Upper Layer Protocol” clearly ties it to the application layer. This further suggests that SCSI has only two protocol layers.

The technical editor wonders how useful it is to say that the architectural model presumes the creation of a new application client for each new request or task management function. It is difficult to see how this formalism serves to produce a better understanding of the real-world usage of SCSI. In fact, other text in the working draft acknowledges that this formalism may not relate to reality at all. If this change were made, it might also be possible to simplify the following statements in 4.3:

> “An application client represents a thread of execution whose functionality is independent of the interconnect and SCSI-3 protocol. In an implementation, that thread could 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-3 protocol."

Is the following 4.2 statement rigorously true?

> "All allusions to a pending command or task management function in this standard are in the application client's frame of reference."

The use of "conventional procedure call" in the following 4.2 statement is at odds with the SAM definitions of procedure call as a modeling mechanism.

> "From the client's standpoint, the behavior of a remote service invoked in this manner is indistinguishable from a conventional procedure call."

If the following two 4.2 statements are true, why are confirmed services defined?

> "In this model, confirmation of successful request or response delivery by the sender is not required. The model assumes that delivery failures will be detected by the client's service delivery port."

The technical editor suspects that "confirmed service" has multiple definitions.

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 Lower Level Interfaces, that reviewed this standard, had the following members:

<<Insert T10 member list>>

Introduction

The SCSI Architecture Model - 3 (SAM-3) standard is divided into seven clauses and one annex:

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 task management functions common to SCSI devices.
Clause 7 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 **INCITS.***:200x**

American National Standard for Information Systems - Information Technology - SCSI Architecture Model - 3 (SAM-3)

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.

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 command descriptor block format and the status values to be returned upon command 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 5.9.6.

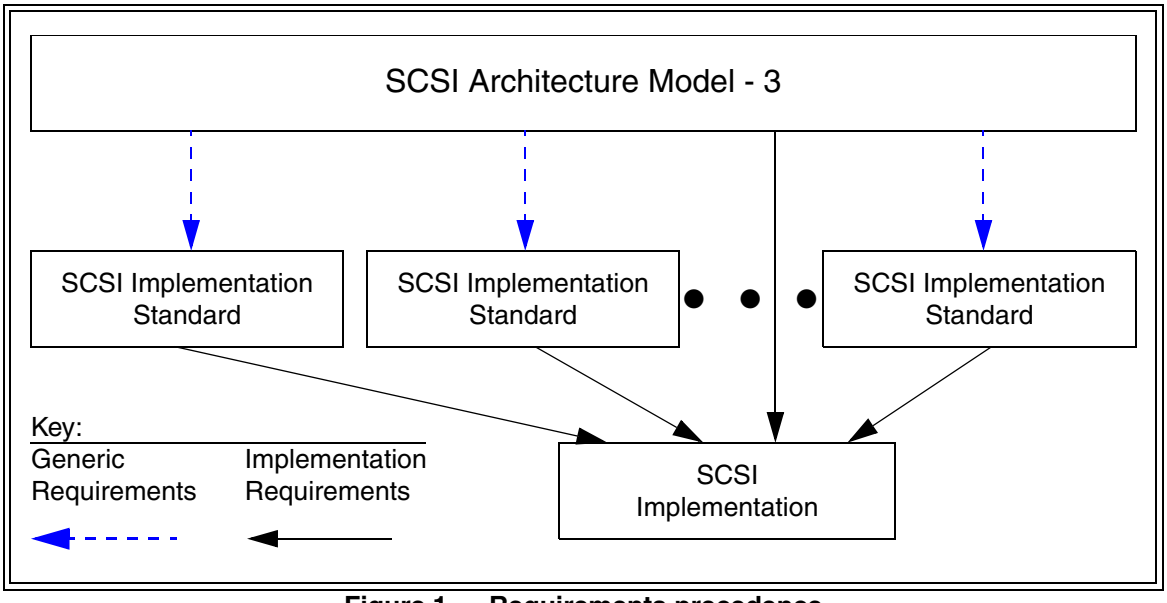


Figure 1 — Requirements precedence

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 require-

ments 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 standards as of the publication of this standard.

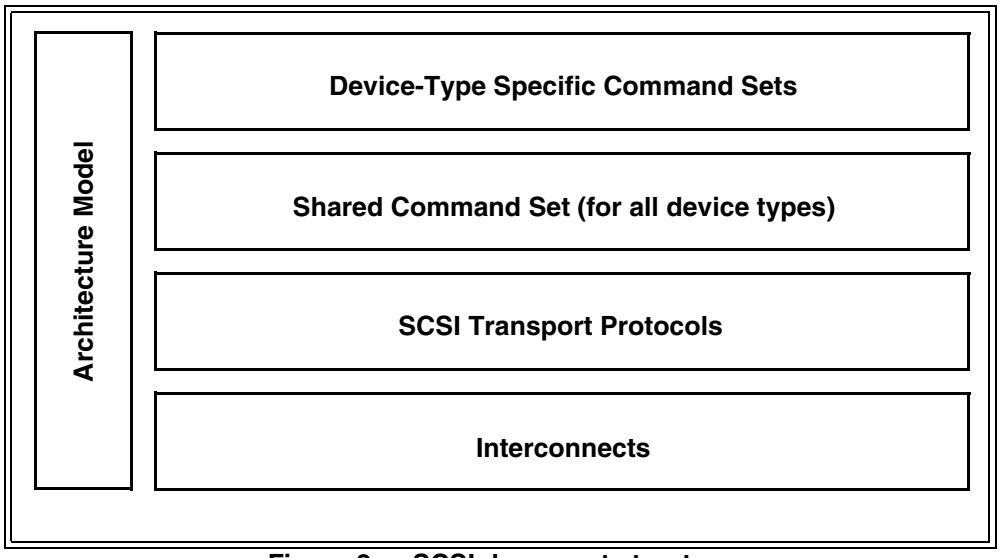


Figure 2 — SCSI document structure

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.

At the time this standard was generated, examples of the SCSI general structure included:

Interconnects:		
Automation/Drive Interface - Physical Layer	ADP	[T10/1556-D]
Fibre Channel Arbitrated Loop - 2	FC-AL-2	[ISO/IEC 14165-122] [ANSI NCITS.332-1999]
Fibre Channel Physical Interfaces	FC-PI	[ANSI INCITS.352-200x]
Fibre Channel Physical Interfaces - 2	FC-PI-2	[T11/1506-D]
Fibre Channel Framing and Signaling Interface	FC-FS	[T11/1331-D]
High Performance Serial Bus		[ANSI IEEE 1394-1995]
High Performance Serial Bus (supplement to ANSI/IEEE 1394-1995)		[ANSI IEEE 1394a-2000]
SCSI Parallel Interface - 2	SPI-2	[ISO/IEC 14776-112] [ANSI X3.302-1999]
SCSI Parallel Interface - 3	SPI-3	[ISO/IEC 14776-113] [ANSI NCITS.336-2000]
SCSI Parallel Interface - 4	SPI-4	[ISO/IEC 14776-114] [ANSI INCITS.362-200x]
SCSI Parallel Interface - 5	SPI-5	[ISO/IEC 14776-115] [T10/1525-D]
Serial Storage Architecture Physical Layer 1	SSA-PH	[ANSI X3.293-1996]
Serial Storage Architecture Physical Layer 2	SSA-PH-2	[ANSI NCITS.307-1998]
Serial Attached SCSI	SAS	[T10/1562-D]
SCSI Transport Protocols:		
Automation/Drive Interface - Transport Protocol	ADT	[T10/1557-D]
Serial Storage Architecture Transport Layer 1	SSA-TL-1	[ANSI X3.295-1996]
Serial Storage Architecture Transport Layer 2	SSA-TL-2	[ANSI NCITS.308-1998]
SCSI-3 Fibre Channel Protocol	FCP	[ISO/IEC 14776-221] [ANSI X3.269-1996]
SCSI Fibre Channel Protocol - 2	FCP-2	[ISO/IEC 14776-222] [ANSI NCITS.350-200x]
SCSI Fibre Channel Protocol - 3	FCP-3	[ISO/IEC 14776-223] [T10/1560-D]
Serial Bus Protocol - 2	SBP-2	[ISO/IEC 14776-232] [ANSI NCITS.325-1999]
Serial Bus Protocol - 3	SBP-3	[ISO/IEC 14776-233] [T10/1467-D]
Serial Storage Architecture SCSI-3 Protocol	SSA-S3P	[ANSI NCITS.309-1998]
SCSI RDMA Protocol	SRP	[T10/1415-D]
Shared Command Sets:		
SCSI-3 Primary Commands	SPC	[ISO/IEC 14776-311] [ANSI X3.301-1997]
SCSI Primary Commands - 2	SPC-2	[ISO/IEC 14776-312] [ANSI NCITS.351-2001]
SCSI Primary Commands - 3	SPC-3	[ISO/IEC 14776-313] [T10/1416-D]

Device-Type Specific Command Sets:

SCSI-3 Block Commands	SBC	[ISO/IEC 14776-321] [ANSI NCITS.306-1998]
SCSI Block Commands - 2	SBC-2	[ISO/IEC 14776-322] [T10/1417-D]
SCSI-3 Stream Commands	SSC	[ISO/IEC 14776-331] [ANSI NCITS.335-2000]
SCSI Stream Commands - 2	SSC-2	[ISO/IEC 14776-332] [T10/1434-D]
SCSI-3 Medium Changer Commands	SMC	[ISO/IEC 14776-351] [ANSI NCITS.314-1998]
SCSI Media Changer Commands - 2	SMC-2	[ISO/IEC 14776-352] [T10/1383-D]
SCSI-3 Multimedia Command Set	MMC	[ANSI X3.304-1997]
SCSI Multimedia Command Set - 2	MMC-2	[ISO/IEC 14776-362] [ANSI NCITS.333-2000]
SCSI Multimedia Command Set - 3	MMC-3	[ISO/IEC 14776-363] [ANSI INCITS.360-2002]
SCSI Multimedia Command Set - 4	MMC-4	[ISO/IEC 14776-364] [T10/1545-D]
SCSI Controller Commands - 2	SCC-2	[ISO/IEC 14776-342] [ANSI NCITS.318-1998]
SCSI Reduced Block Commands	RBC	[ISO/IEC 14776-326] [ANSI NCITS.330-2000]
SCSI-3 Enclosure Services Commands	SES	[ISO/IEC 14776-371] [ANSI NCITS.305-1998]
SCSI Enclosure Services Commands - 2	SES-2	[ISO/IEC 14776-372] [T10/1559-D]
SCSI Specification for Optical Card Reader/Writer	OCRW	[ISO/IEC 14776-381]
Object-based Storage Devices Commands	OSD	[T10/1355-D]
SCSI Management Server Commands	MSC	[T10/1528-D]
Automation/Drive Interface - Commands	ADC	[T10/1558-D]

Architecture Model:

SCSI-3 Architecture Model	SAM	[ISO/IEC 14776-411] [ANSI X3.270-1996]
SCSI Architecture Model - 2	SAM-2	[ISO/IEC 14776-412] [T10/1157-D]
SCSI Architecture Model - 3	SAM-3	[ISO/IEC 14776-413] [T10/1561-D]

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

ISO/IEC 14776-312, SCSI Primary Commands - 2 (SPC-2) [ANSI NCITS.351-2001]

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 14776-313, SCSI Primary Commands - 3 (SPC-3) [T10/1416-D]

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 3.1.4, 4.10, and 7.5.4).

3.1.2 additional sense code: A combination of the ADDITIONAL SENSE CODE and ADDITIONAL SENSE CODE QUALIFIER fields in the sense data (see 3.1.104 and SPC-2).

3.1.3 application client: An object that is the source of SCSI commands.

3.1.4 auto contingent allegiance (ACA): One of the possible conditions of a task set following the return of a CHECK CONDITION status. See 5.9.1.

3.1.5 blocked task state: When in this state a task is prevented from completing due to a CA or ACA condition.

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

3.1.7 byte: An 8-bit construct.

3.1.8 call: The act of invoking a procedure.

3.1.9 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.

3.1.10 client: An entity that requests a service from a server.

3.1.11 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 3.1.35), in a procedure input data parameter (see 3.6.2), in a procedure output data parameter, or in a procedure result.

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

3.1.13 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 up to 16 bytes or a variable length of between 12 and 260 bytes.

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

3.1.15 completed task: A 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.

3.1.16 confirmation: A response returned to a client or server that signals the completion of a service request.

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

3.1.18 contingent allegiance (CA): One of the possible conditions of a task set following the return of a CHECK CONDITION status. See 5.9.1.

3.1.19 Control mode page: The mode page that identifies the settings of several device server behaviors that may be of interest to an application client or may be changed by an application client. Fields in the Control mode page are referenced by name in this standard and SPC-2 contains a complete definition of the Control mode page.

3.1.20 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 should define the SCSI transport protocol specific conditions under which a task is considered a current task.

3.1.21 dependent logical unit: A logical unit that is addressed via some other logical unit(s) in a hierarchical logical unit structure (see 3.1.38), also a logical unit that is at a higher numbered level in the hierarchy than the referenced logical unit (see 4.13).

3.1.22 device identifier: A term used by previous versions of this standard (see Annex B).

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

3.1.24 device server: An object within a logical unit that processes SCSI tasks according to the requirements for task management described in clause 7.

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

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

3.1.27 domain: An I/O system consisting of a set of SCSI devices that interact with one another by means of a service delivery subsystem.

3.1.28 dormant task state: When in this state a task is prevented from entering the enabled task state (see 3.1.29) due to the presence of certain other tasks in the task set.

3.1.29 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.

3.1.30 ended command: A command that has completed or aborted.

3.1.31 faulted initiator port: The SCSI initiator port to which a CHECK CONDITION status was returned. The faulted initiator port condition is cleared when the CA or ACA condition resulting from the CHECK CONDITION status is cleared.

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

3.1.33 faulting command: A command that completed with a status of CHECK CONDITION.

3.1.34 faulting task: A task that has completed with a status of CHECK CONDITION.

3.1.35 field: A group of one or more contiguous bits, part of a larger structure such as a CDB (see 3.1.13) or sense data (see 3.1.104).

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

3.1.37 hard reset: A SCSI device action in response to a reset event in which SCSI target port performs the operations described in 5.9.6.

3.1.38 hierarchical logical unit: An inverted tree structure for forming and parsing logical unit numbers (see 3.1.60) containing up to four addressable levels (see 4.13).

3.1.39 I_T nexus: A nexus between a SCSI initiator port and a SCSI target port (see 4.11).

3.1.40 I_T_L nexus: A nexus between a SCSI initiator port, a SCSI target port, and a logical unit (see 4.11).

3.1.41 I_T_L_Q nexus: A nexus between a SCSI initiator port, a SCSI target port, a logical unit, and a tagged task (see 4.11).

3.1.42 I_T_L_x nexus: Either an I_T_L nexus or an I_T_L_Q nexus (see 4.11).

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

3.1.44 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.

3.1.45 implementation option: An option whose actualization within an implementation is at the discretion of the implementor.

3.1.46 initiator: A term used by previous versions of this standard (see Annex B).

3.1.47 initiator device name: A SCSI device name of a SCSI initiator device (see 4.7.1).

3.1.48 initiator identifier: A term used by previous versions of this standard (see Annex B).

3.1.49 initiator port identifier: A value by which a SCSI initiator port is referenced within a domain. See 4.7.1.

3.1.50 initiator port name: A SCSI port name (see 3.1.93) of a SCSI initiator port or of a SCSI target/initiator port when operating as a SCSI initiator port. See 4.7.1.

3.1.51 interconnect subsystem: One or more interconnects that appear as a single path for the transfer of information between SCSI devices in a domain.

3.1.52 in transit: Information that has been sent to a remote entity but not yet received.

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

3.1.54 linked CDB: A CDB with the LINK bit in the CONTROL byte set to one.

3.1.55 linked command: One in a series of SCSI 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_x nexus, and all, except the last, have the LINK bit in the CDB CONTROL byte set to one.

3.1.56 logical unit: A SCSI target device object, containing a device server and task manager, that implements a device model and manages tasks to process SCSI commands sent by an application client. See 4.8.

3.1.57 logical unit reset: A logical unit action in response to a logical unit reset event in which the logical unit performs the operations described in 5.9.7.

3.1.58 logical unit reset event: An event that triggers a logical unit reset from a logical unit as described in 5.9.7.

3.1.59 logical unit inventory: The list of the logical unit numbers reported by a REPORT LUNS command (see SPC-2).

3.1.60 logical unit number (LUN): A 64-bit identifier for a logical unit.

3.1.61 logical unit option: An option pertaining to a logical unit, whose actualization is at the discretion of the logical unit implementor.

3.1.62 lower level protocol (LLP): A protocol used to carry the information representing upper level protocol transactions.

3.1.63 media information: Information stored within a SCSI device that is non-volatile (retained through a power cycle) and accessible to a SCSI initiator device through the processing of SCSI commands.

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

3.1.65 nexus: A relationship between two SCSI devices, and the SCSI initiator port and SCSI target port objects within those SCSI devices. See 4.11.

3.1.66 non-faulted initiator port: A SCSI initiator port that is not a faulted initiator port (see 3.1.31).

3.1.67 object: An architectural abstraction or container that encapsulates data types, services, or other objects that are related in some way.

3.1.68 peer-to-peer protocol service: A service used by an upper level protocol implementation to exchange information with its peer.

3.1.69 peer entities: Entities within the same layer.

3.1.70 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.

3.1.71 port: Synonymous with SCSI port (see 3.1.91).

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

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

3.1.74 protocol option: A function whose definition within a SCSI transport protocol standard is optional.

3.1.75 queue: The arrangement of tasks within a task set (see 3.1.125), usually according to the temporal order in which they were created.

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

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

3.1.78 request: A transaction invoking a service.

3.1.79 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.

3.1.80 request-confirmation transaction: An interaction between a pair of cooperating entities, consisting of a request for service submitted to an entity followed by a response from the entity confirming request completion.

3.1.81 reset event: A SCSI transport protocol specific event that triggers a hard reset from a SCSI device as described in 5.9.6.

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

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

3.1.84 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.

3.1.85 SCSI device identifier: Synonymous with SCSI port identifier (see 3.1.92).

3.1.86 SCSI device name: A name (see 3.1.64) 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 4.7.6). The SCSI device name may be made available to other SCSI devices or SCSI ports in that SCSI domain in SCSI transport protocol specific ways.

3.1.87 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.

3.1.88 SCSI identifier: A term used by previous versions of this standard (see Annex B).

3.1.89 SCSI initiator device: A SCSI device containing application clients and SCSI initiator ports that originate device service and task management requests to be processed by a SCSI target device. When used this term refers to SCSI initiator devices or SCSI target/initiator devices that are using the SCSI target/initiator port as a SCSI initiator port.

3.1.90 SCSI initiator port: A SCSI initiator device object that acts as the connection between application clients and the service delivery subsystem through which requests and responses 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.

3.1.91 SCSI port: A 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 either a SCSI initiator port (see 3.1.90) or a SCSI target port (see 3.1.100).

3.1.92 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 3.1.49) or a target port identifier (see 3.1.116).

3.1.93 SCSI port name: A name (see 3.1.64) of a SCSI port that is world wide unique within the SCSI transport protocol of the SCSI domain of that SCSI port (see 4.7.7). The name may be made available to other SCSI devices or SCSI ports in that SCSI domain in SCSI transport protocol specific ways.

3.1.94 SCSI transport protocol layer: The protocol and services used by a SCSI application layer to transport data representing a SCSI application protocol transaction.

3.1.95 SCSI transport protocol service confirmation: A signal from the lower level SCSI transport protocol layer notifying the upper layer that a SCSI transport protocol service request has completed.

3.1.96 SCSI transport protocol service indication: A signal from the lower level SCSI transport protocol layer notifying the upper level that a SCSI transport protocol transaction has occurred.

3.1.97 SCSI transport protocol service request: A call to the lower level SCSI transport protocol layer to begin a SCSI transport protocol service transaction.

3.1.98 SCSI transport protocol service response: A reply from the upper level protocol layer in response to a SCSI transport protocol service indication.

3.1.99 SCSI target device: A SCSI device containing logical units and SCSI target ports that receives device service and task management requests for processing. When used this term refers to SCSI target devices or SCSI target/initiator devices that are using the SCSI target/initiator port as a SCSI target port.

3.1.100 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 the service delivery subsystem through which requests 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.

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

3.1.102 SCSI target/initiator port: A SCSI device resident object that has all the characteristics of a SCSI target port and a SCSI initiator port.

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

3.1.104 sense data: Data returned to an application client as a result of an autosense operation, asynchronous event report, or REQUEST SENSE command (see 5.9.4). Fields in the sense data are referenced by name in this standard. See SPC-2 for a complete sense data format definition.

3.1.105 sense key: A field in the sense data (see 3.1.104 and SPC-2).

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

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

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

3.1.109 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.

3.1.110 service delivery transaction: A request or response sent through the service delivery subsystem.

3.1.111 signal: (n) A detectable asynchronous event possibly accompanied by descriptive data and parameters. (v) The act of generating such an event.

3.1.112 standard INQUIRY data: Data returned to an application client as a result of an INQUIRY command. Fields in the standard INQUIRY data are referenced by name in this standard and SPC-2 contains a complete definition of the standard INQUIRY data format.

3.1.113 target: A term used by previous versions of this standard (see Annex B).

3.1.114 target device name: A SCSI device name (see 3.1.86) of a SCSI target device. See 4.7.2.

3.1.115 target identifier: A term used by previous versions of this standard (see Annex B).

3.1.116 target port identifier: A value by which a SCSI target port is referenced within a domain. See 4.7.2.

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

3.1.118 target/initiator device name: A SCSI device name (see 3.1.86) of a SCSI target/initiator device. See 4.7.3.

3.1.119 **task:** An object within the logical unit representing the work associated with a command or a group of linked commands.

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

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

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

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

3.1.124 task router: An object in a SCSI target port that routes commands and task management functions between the service delivery subsystem (see 3.1.109) and the appropriate logical unit's task manager (see 3.1.123).

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

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

3.1.127 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.

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

3.1.129 unlinked command: A SCSI command having the LINK bit set to zero in the CDB CONTROL byte.

3.1.130 upper level protocol (ULP): An application-specific protocol processed through services provided by a lower level protocol.

3.1.131 **wakeup:** A SCSI target port returning from the sleep power condition to the active power condition (see SPC-3).

3.1.132 wakeup event: An event that triggers a wakeup from a SCSI target port as described in SPC-3.

3.1.133 well known logical unit: A logical unit that only performs specific functions. See 4.9.9. Well known logical units allow an application client to issue requests to receive and manage specific information usually relating to a SCSI target device.

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

3.2 Acronyms

ACA	Auto Contingent Allegiance (see 3.1.4)
AER	Asynchronous Event Reporting (see 5.9.4.2)
CA	Contingent Allegiance (see 3.1.18)
CDB	Command Descriptor Block (see 3.1.13)
LLP	Lower Level Protocol (see 3.1.62)
LUN	Logical Unit Number (see 3.1.60)
MMC-2	SCSI Multi-Media Commands -2 (see 1.3)
n/a	Not Applicable
RAID	Redundant Array of Independent Disks
SBC	SCSI-3 Block Commands (see 1.3)
SCSI	The architecture defined by the family of standards described in 1.3
SPI-4	SCSI Parallel Interface -4 (see 1.3)
SPC-2	SCSI Primary Commands -2 (see 1.3)
SPC-3	SCSI Primary Commands -3 (see 1.3)
SSC	SCSI-3 Stream Commands (see 1.3)
ULP	Upper Level Protocol (see 3.1.130)
VPD	Vital Product Data (see SPC-2)
W-LUN	Well known logical unit number (see 3.1.134)

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, procedures, parameters or discrete states are capitalized or set in bold type. Names of fields are identified using small capital letters (e.g., NACA bit).

Callable procedures are identified by a name in bold type, such as **Execute Command** (see clause 5). Names of procedural 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 23 (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

Digits 0 through 9 in the text of this standard that are not immediately followed by lower-case "b" or "h" are decimal values. Digits 0 and 1 immediately followed by lower case "b" are binary values. Digits 0 through 9 and the upper case letters "A" through "F" immediately followed by lower-case "h" are hexadecimal values.

Large numbers are separated by spaces (e.g., 12 345, not 12,345).

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 3, 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.

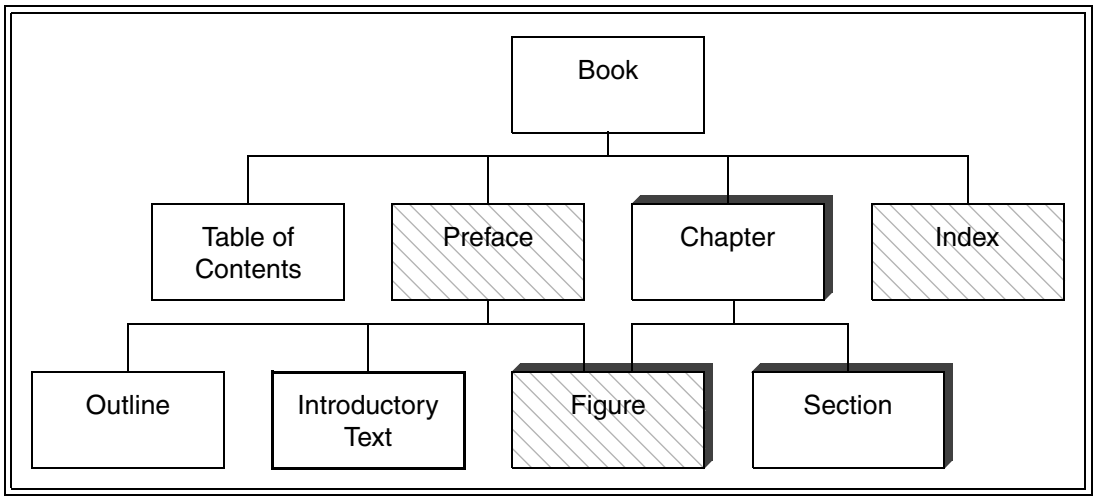


Figure 3 — Example hierarchy diagram

In the corresponding hierarchy diagram, labeled 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.

3.6.2 Notation for procedures and functions

In this standard, the model for functional interfaces between entities is the callable procedure. 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 or function.

Procedure Name: A descriptive name for the function to be performed.

Input-1, Input-2, …: A comma-separated list of names identifying caller-supplied input data parameters.

Output-1, Output-2, …: A comma-separated list of names identifying output data parameters to be returned by the procedure.

"[…]": Brackets enclosing optional or conditional parameters and arguments.

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

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

Where:

Found = Flag
Flag, if set, indicates that a matching item was located.

Input Arguments:

Pattern = … /* Definition of **Pattern** parameter */
Parameter containing the search pattern.

Item List = Item<NN> /* Definition of **Item List** as an array of NN **Item** parameters*/
Contains the items to be searched for a match.

Output Arguments:

Item Found = Item … /* Item located by the search procedure */
This parameter is only returned if the search succeeds.

3.6.3 Notation for state diagrams

All state diagrams use the notation shown in figure 4.

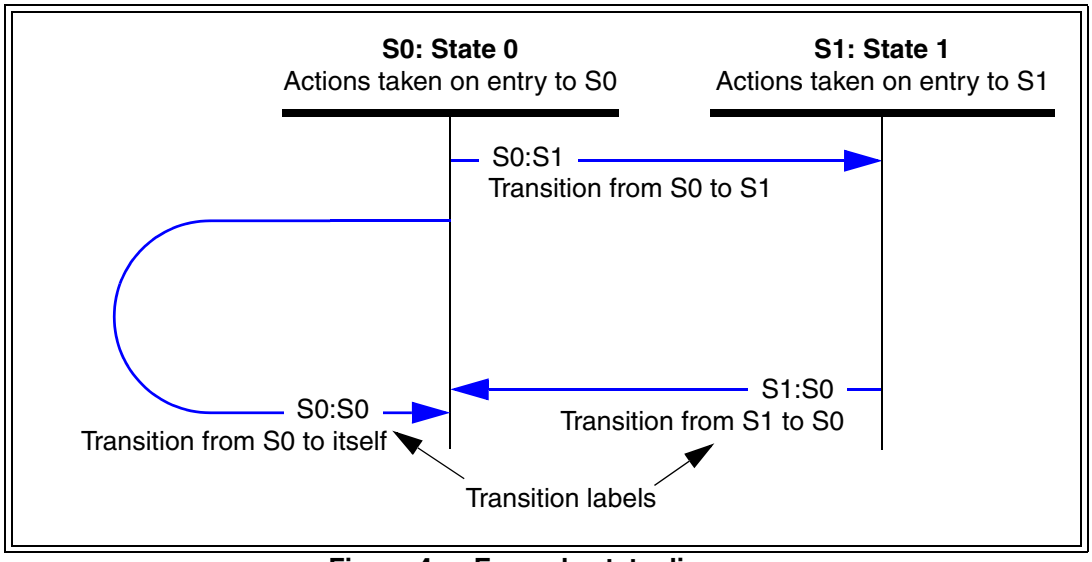


Figure 4 — Example state diagram

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 4 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) Identify areas for developing standards and provide a common reference for maintaining consistency among related standards so that independent teams of experts may work productively and independently on the development of standards within each functional area; and
c) 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 such as 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 the architecture model.

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 3.1.67), 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, such as an interconnect, are defined by SCSI, while others, such as a task, are needed to understand the functioning of SCSI but have implementation definitions outside the scope of SCSI. That is, although such objects exhibit well-defined and observable behaviors, they do not exist as separate physical elements. An object may be a single numeric parameter, such as a logical unit number, 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 6.

Requirements that apply to each SCSI transport protocol standard are specified in the SCSI transport protocol service model described in 5.4 and 6.10. 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 5. 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.

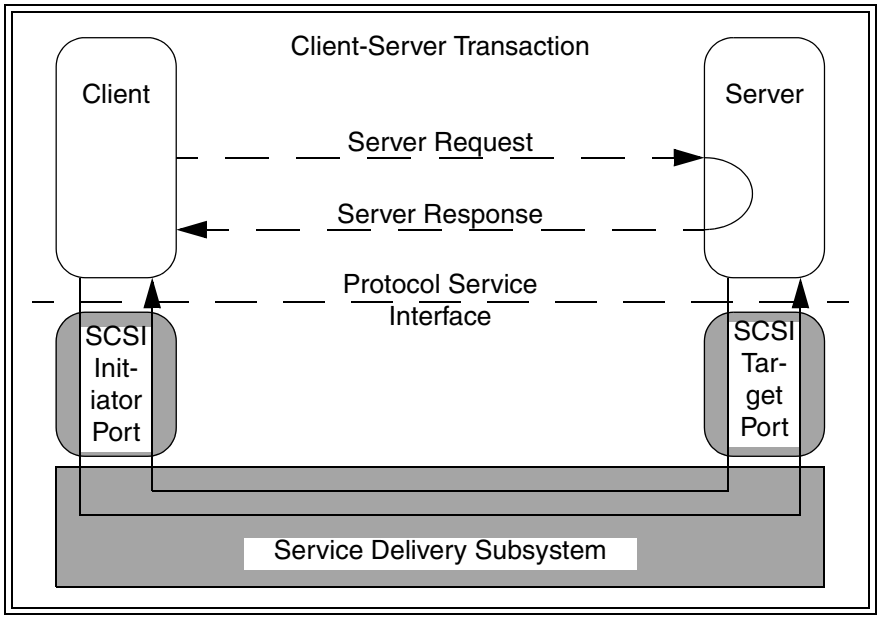


Figure 5 — Client-Server model

A client-server transaction is represented as a remote procedure call with inputs supplied by the caller (the client). The procedure is processed by the server and returns outputs and a procedure status. A client directs requests to a remote server, via the client's 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 an error-free copy of the request or response between sender and receiver. A failure notification indicates that a condition has been detected, such as 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 its service delivery subsystem 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 logical unit state. All references to a pending command or task management function in this standard are from the application client's point of view.

Client-server relationships are not symmetrical. A client may only originate requests for service. A server may only respond to such requests. The client calls the server-resident procedure and waits for completion. From the client's point of view, the behavior of a remote service invoked in this manner is indistinguishable from a conventional procedure call. 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 client's SCSI port or within service delivery subsystem.

4.3 The SCSI client-server model

As shown in figure 6, 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 SCSI commands such as reading from or writing to the media. Each pending SCSI command or series of linked commands defines a unit of work to be performed by the logical unit. Each unit of work is represented within the SCSI target device by a task that may be externally referenced and controlled through requests issued to the task manager.

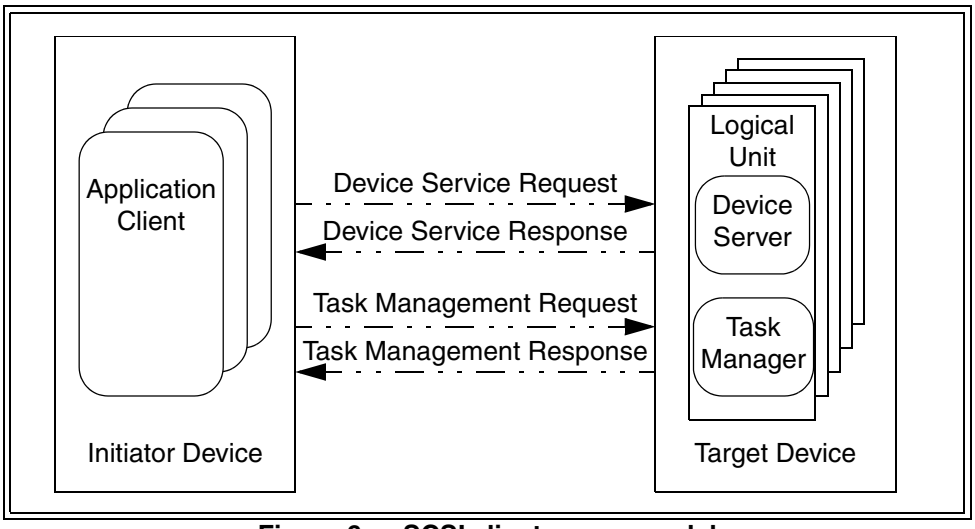


Figure 6 — SCSI client-server model

All requests originate from application clients residing within a SCSI initiator device. An application client is independent of the interconnect and SCSI transport protocol. In an implementation, 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. In the architecture model, an application client creates one or more application client tasks each of which issues a single SCSI command, series of linked SCSI commands, or task management function. Application client tasks are considered to be 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. Consequently, there is one application client task for each pending command, series of linked commands, or task management request.

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 SCSI 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 sequence 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 command completion 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 single command see 5.8.1. For an example of linked command processing see 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 6.11.

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 7, 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, and related information. A SCSI device contains application clients or device servers or both and the infrastructure to support them.

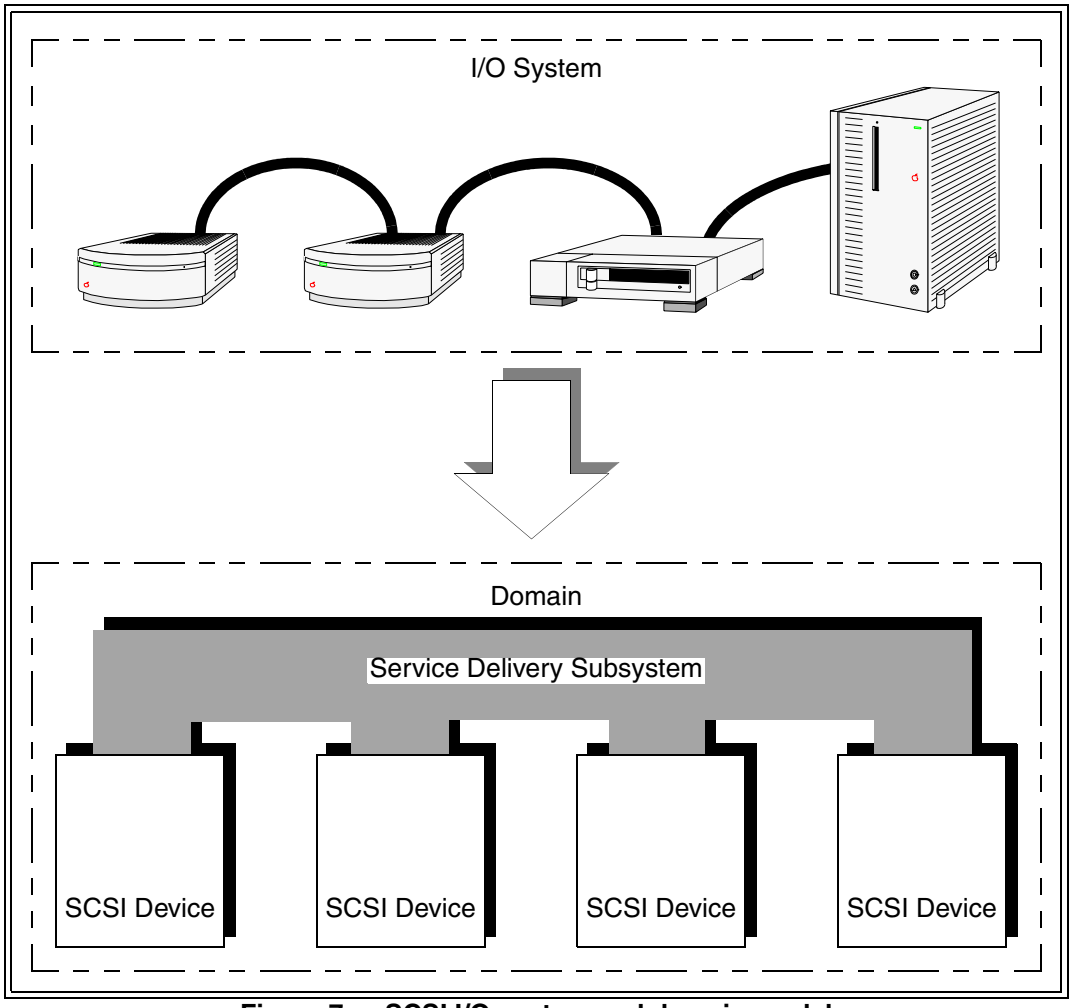


Figure 7 — SCSI I/O system and domain model

Figure 8 shows the main functional components of the SCSI domain. This standard defines these components in greater detail.

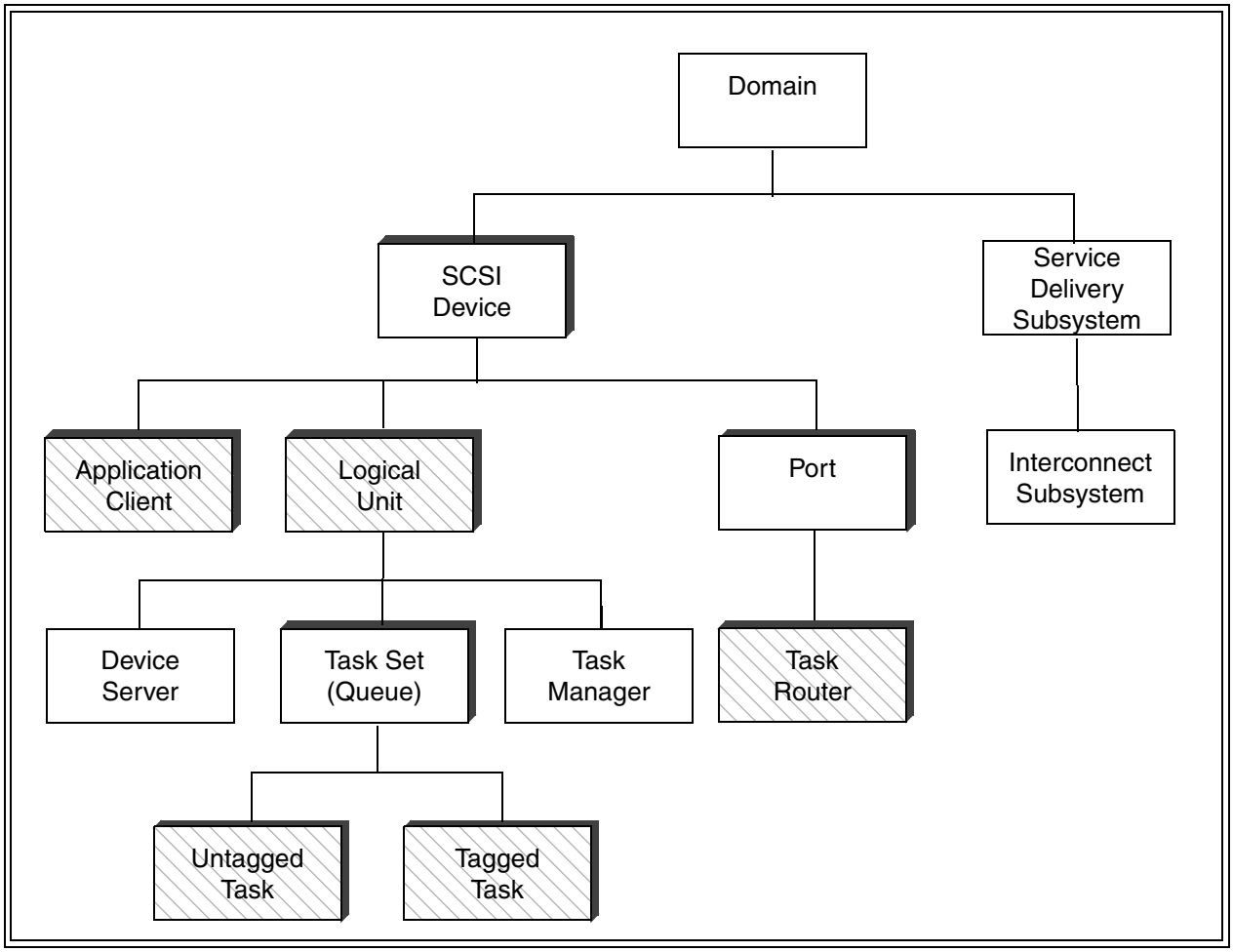


Figure 8 — Overall SCSI domain model

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 9).

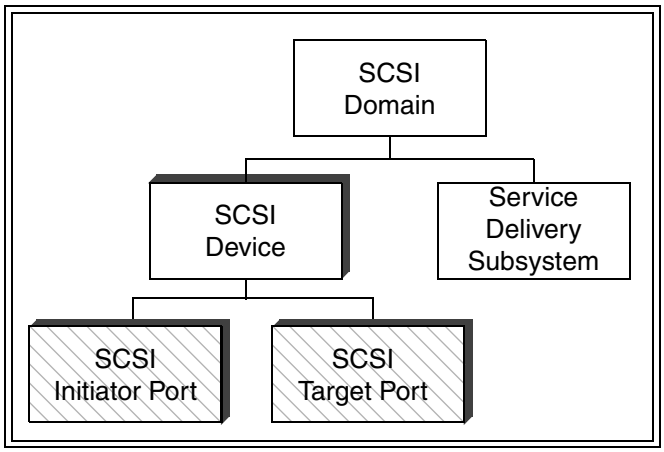


Figure 9 — SCSI domain model

A SCSI device is an object that originates or services SCSI commands. As described in 4.7, when a SCSI device originates a command it is called a SCSI initiator device and that command is transmitted through a SCSI initiator port or a SCSI target/initiator port. A SCSI device containing logical units that service commands is called a SCSI target device and receives commands through a SCSI target port or a SCSI target/initiator port. The service delivery subsystem connects all the SCSI ports in the SCSI domain, providing a mechanism through which application clients and device servers communicate (see 4.6). The boundaries of a SCSI domain are established by the system implementor, within the constraints of a specific SCSI transport protocol and interconnect standards.

4.6 The service delivery subsystem

4.6.1 The service delivery subsystem object

The service delivery subsystem connects SCSI ports (see 3.1.91) and is composed of an interconnect subsystem (see figure 10).

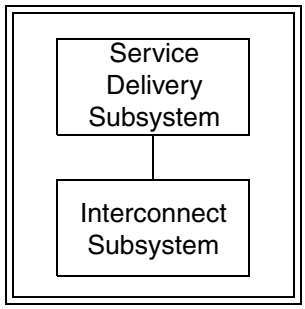


Figure 10 — Service delivery subsystem model

The interconnect subsystem is a set of one or more interconnects that appear to a client or server as a single path for the transfer of requests, responses, and data 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. In this model, the data comprising an outgoing request is sent in a single package containing all the information required to process the remote procedure call. Similarly, an incoming server response is returned in a package enclosing the output data and status. The request or response package is sent when it is passed to the SCSI port for transmission; it is in transit until delivered and received when it has been forwarded to the receiver via the destination SCSI device's SCSI port.

4.6.2 Synchronizing client and server states

The client is usually informed of changes in server state through the arrival of server responses. In the architecture model 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. For example, 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.

Some 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.

The model 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 6.10 and 5.4, it is assumed that the service delivery 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

In this standard, 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 occasionally 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 SCSI commands with the ORDERED attribute to a logical unit as described in clause 7). 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. Similarly, the SCSI initiator device acquires knowledge about the state of pending commands and task management functions and may subsequently take action based on the nature and sequence of SCSI target device responses (e.g., if the SCSI initiator device aborts a command whose completion response is in transit and the abort response is received out of order, the SCSI initiator device may incorrectly conclude that no further responses are expected from that command).

The manner in which ordering constraints are established is vendor specific. An implementation may choose to 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 SCSI architecture model assumes in-order delivery to be a property of the service delivery subsystem. This assumption is made to simplify the description of behavior and does not constitute a requirement. This specification makes no assumption about, or places any requirement on the ordering of requests or responses between tasks or task management functions received from different SCSI initiator ports.

4.7 SCSI devices

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 SCSI commands and task management requests (see 4.7.1). A SCSI target device contains at least one SCSI target port and is capable of processing SCSI commands and task management requests (see 4.7.2). A SCSI target/initiator device contains at least one SCSI target/initiator port and is capable of originating and processing SCSI commands and task management requests (see 4.7.3). To be functional, a SCSI domain needs 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.

4.7.1 SCSI initiator device

A SCSI initiator device (see figure 11) contains:

a) Zero or more initiator device names;
b) One or more SCSI initiator ports each containing an initiator port identifier and an optional initiator port name; and
c) One or more application clients.

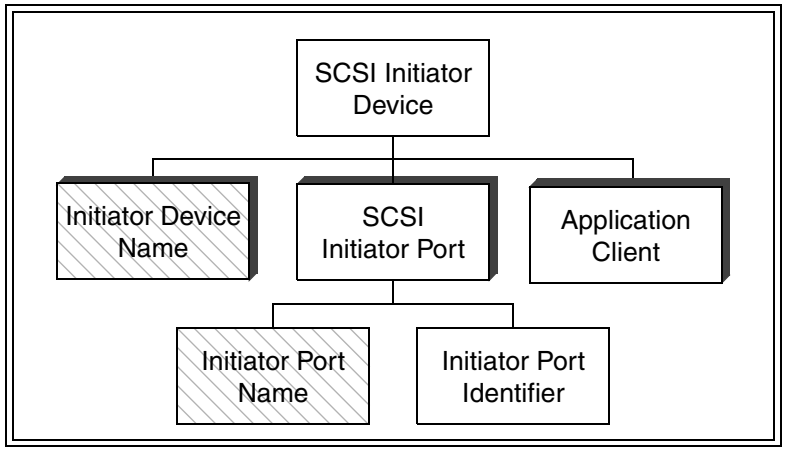


Figure 11 — SCSI initiator device model

An initiator port identifier is a value that is the SCSI port identifier (see 4.7.4) for an initiator port.

An initiator device name is a name (see 3.1.64) that is a SCSI device name (see 4.7.6) for a SCSI initiator device. A SCSI initiator device shall have no more than one initiator device name for each supported SCSI transport protocol. A SCSI transport protocol standard may place additional requirements on initiator device names.

An initiator port name is a name (see 3.1.64) that is the SCSI port name (see 4.7.7) for the 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.

4.7.2 SCSI target device

A SCSI target device (see figure 12) contains:

a) Zero or more target device names;
b) One or more SCSI target ports each containing a task router, SCSI target port identifier, and an optional target port name; and
c) One or more logical units.

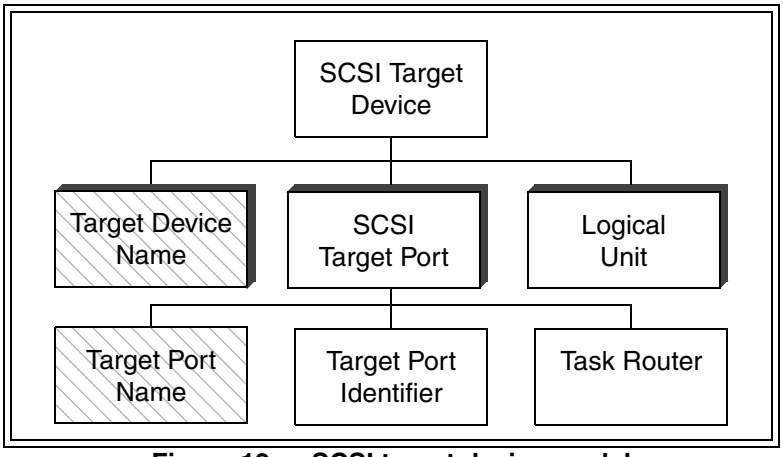


Figure 12 — SCSI target device model

A SCSI target port identifier is a value that is a SCSI port identifier (see 4.7.4) for a SCSI target port.

A target device name is a name (see 3.1.64) that is a SCSI device name (see 4.7.6) for a SCSI target device. A SCSI target device shall have no more than one target device name for each supported SCSI transport protocol. A SCSI transport protocol standard may place additional requirements on target device names.

A target port name is a name (see 3.1.64) that is the SCSI port name (see 4.7.7) for the target port. A SCSI transport protocol standard may place additional requirements on target port names.

A task router routes commands and task management functions between the service delivery subsystem and the appropriate logical unit's task manager (see 4.7.5).

A logical unit is the object to which SCSI commands are addressed. One of the logical units within the SCSI target device shall be accessed using the logical unit number zero. See 4.8 for a description of the logical unit.

4.7.3 SCSI target/initiator device

A SCSI target/initiator device (see figure 13) contains:

a) Zero or more target/initiator device names;
b) One or more SCSI target/initiator ports each containing a task router, target port identifier, an initiator port identifier, an optional target port name and an optional initiator port name;
c) One or more logical units; and
d) One or more application clients.

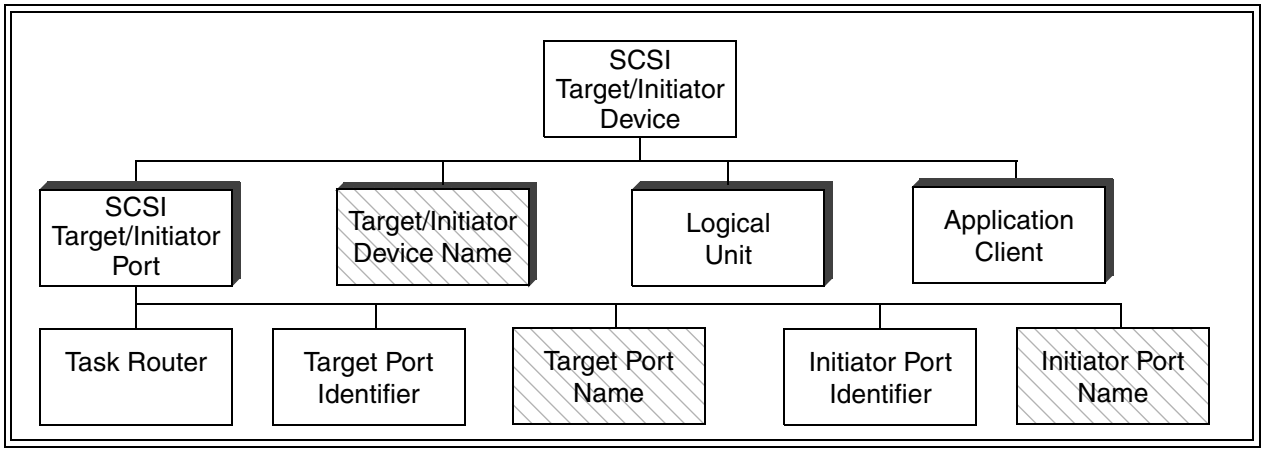


Figure 13 — SCSI target/initiator device model

The target port identifier and the initiator port identifier are values containing a SCSI port identifier (see 4.7.4) for a SCSI target/initiator port. The target port identifier and the initiator port identifier may or may not be identical.

A target/initiator device name is a name (see 3.1.64) that is a SCSI device name (see 4.7.6) for a SCSI target/initiator device. A SCSI target/initiator device shall have no more than one target/initiator device name for each supported SCSI transport protocol. A SCSI transport protocol standard may place additional requirements on target/initiator device names.

The target port name and initiator port name are names (see 3.1.64) that are the SCSI port name (see 4.7.7) for the target/initiator port when operating as a target port and initiator port, respectively. The target port name and the initiator port name may or may not be identical. A SCSI transport protocol standard may place additional requirements on target port names and initiator port names.

When the SCSI target/initiator device is operating as a SCSI target device a task router routes the commands and task management functions between the service delivery subsystem and the appropriate logical unit (see 4.7.5).

A logical unit is the object to which SCSI commands are sent. One of the logical units within the SCSI target/initiator device shall be accessed using the logical unit number zero. See 4.8 for a description of the logical unit.

When the SCSI target/initiator device is operating as a SCSI initiator device an application client is the source of commands and task management functions.

4.7.4 SCSI port identifier

The SCSI port identifier is equivalent to SCSI identifier. The SCSI port identifier object represents either an initiator port identifier for a SCSI initiator port, or a target port identifier for a SCSI target port. SCSI port identifier is used when either a SCSI initiator port or SCSI target port is applicable or when other context in the description identifies the SCSI initiator port or SCSI target port usage.

4.7.5 SCSI task router

The task router routes tasks and task management functions to the selected logical unit. Any task that is sent to a logical unit that is not known to the task router is handled as described in 5.9.3. Any task management function that is not sent to a specific logical unit shall be broadcast to all logical units known to the task router.

4.7.6 SCSI device name

A SCSI device name is an optional name (see 3.1.64) 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. A SCSI device may have more than one name if that device has SCSI ports in different SCSI transport protocol domains. A SCSI device shall have no more than one name for each supported SCSI transport protocol. 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 if the SCSI device has SCSI ports in a SCSI domain of that SCSI transport protocol. The SCSI device name may be made available to other SCSI devices or SCSI ports in a given SCSI domain in SCSI transport protocol specific ways.

4.7.7 SCSI port name

A SCSI port name is an optional name (see 3.1.64) of a SCSI port that is world wide unique within the SCSI transport protocol of the SCSI domain of that SCSI port. A SCSI port may have at most one name. A SCSI port name shall never change and may be used to persistently identify a SCSI initiator port or SCSI target port in contexts similar to those where a SCSI port identifier (see 4.7.4) may be used.

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

4.8 Logical units

A logical unit (see figure 14) contains:

a) A logical unit number;
   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;
b) A device server;
c) A task manager; and
d) One or more task sets each of which may contain zero or more untagged tasks or a combination of zero or more tagged tasks and zero or more untagged tasks.

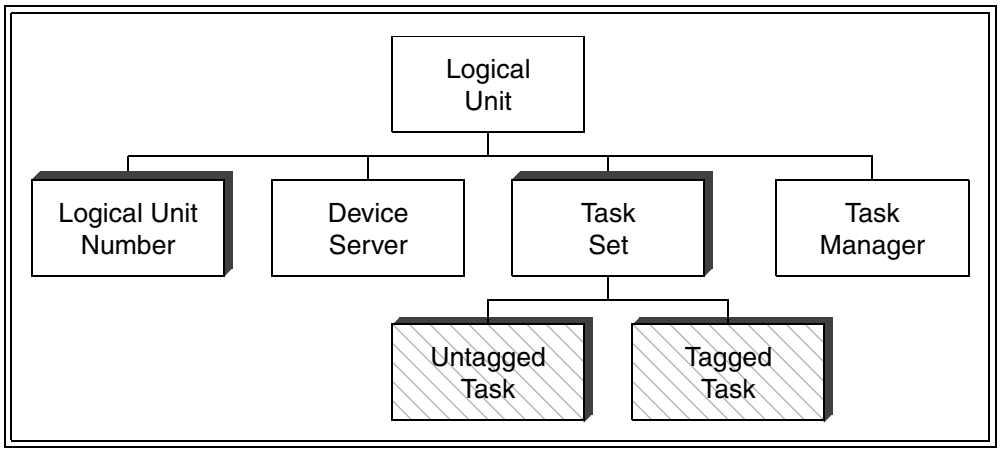


Figure 14 — Logical unit model

A logical unit number is a field (see 4.9) containing up to 64 bits that 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 dependent logical units in its composition, all logical unit numbers within the scope of the SCSI target device shall have the format described in 4.9.4. Otherwise, the logical unit numbers should have the format described in 4.9.3.

A device server is the object that processes the operations requested by the received commands.

The task manager controls the sequencing of one or more tasks within a logical unit. The task manager also carries out the task management functions specified in clause 6. There is one task manager per logical unit.

The order in which task management requests are processed is not specified by this standard. This standard does not require in-order delivery of such requests, as defined in 4.6.3, or processing by the task manager in the order received. To guarantee the processing order of task management requests referencing a specific logical unit, a SCSI initiator port should not have more than one such request pending to that logical unit.

A task set is composed of zero or more untagged tasks or a combination of zero or more tagged tasks and zero or more untagged tasks. See 4.10 for additional restrictions on the untagged tasks and tagged tasks in a task set.

Task (see 4.10) refers to either a tagged task or an untagged task. The interactions among the tasks in a task set are determined by the requirements for task set management specified in clause 7 and the CA or ACA requirements specified in 5.9.1. 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-2).

4.9 Logical unit numbers

4.9.1 Logical unit numbers overview

All logical unit number formats described in this standard are hierarchical in structure even when only a single level in that hierarchy is used. The HISUP bit shall be set to one in the standard INQUIRY data (see SPC-2) when any logical unit number format described in this standard is used. Non-hierarchical formats are outside the scope of this standard.

4.9.2 LUN 0 address

All SCSI devices shall accept LUN 0 as a valid address. For SCSI devices that support the hierarchical addressing model the LUN 0 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.

To address the LUN 0 of a SCSI device the peripheral device address method shall be used.

4.9.3 Single level logical unit number structure

When the single level subset format is used, the HISUP bit shall be set to one in the standard INQUIRY data (see SPC-2) returned by the logical unit with the logical unit number zero.

If a SCSI target device contains 256 or fewer logical units, none of which are dependent logical units (see 4.13) or extended addressing logical units (see 4.9.8), then its logical unit numbers shall have the format shown in table 1, that is a single level subset of the format described in 4.13.

Table 1 — Single level logical unit number structure for 256 or fewer logical units

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

All logical unit number structure fields shall be zero except the SINGLE LEVEL LUN field (see table 1). The value in the SINGLE LEVEL LUN field shall be between 0 and 255. The 00b in the ADDRESS METHOD field and the 00h in the BUS IDENTIFIER field specify addressing for a logical unit at the current level (see 4.9.4).

If a SCSI target device contains more than 256 and less than 16 385 logical units, none of which are dependent logical units (see 4.13) or extended addressing logical units (see 4.9.8), then its logical unit numbers that are greater than 255 shall have the format shown in table 2, that is a single level subset of the format described in 4.13. Logical unit numbers between 0 and 255 should have the format shown in table 1 but may have the format shown in table 2.

Table 2 — Single level logical unit number structure for 257 to 16 384 logical units

Bit / Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (01b)		(MSB)					
1				SINGLE LEVEL LUN (0100h to 3FFFh, inclusive)				(LSB)
2	(MSB)			Null second level LUN (0000h)				
3								(LSB)
4	(MSB)			Null third level LUN (0000h)				
5								(LSB)
6	(MSB)			Null forth level LUN (0000h)				
7								(LSB)

All logical unit number structure fields shall be zero except the SINGLE LEVEL LUN field (see table 2). The value in the SINGLE LEVEL LUN field shall be between 256 and 16 383. The 01b in the ADDRESS METHOD field specifies flat address space addressing for a logical unit at the current level (see 4.9.7).

4.9.4 Eight byte logical unit number structure

The eight byte logical unit number structure (see table 4) allows up to four levels of SCSI devices to be addressed under a single SCSI target device. Each level shall use byte 0 and byte 1 to define the address and/or 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 layer then the current layer 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 3 and figure 15).

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

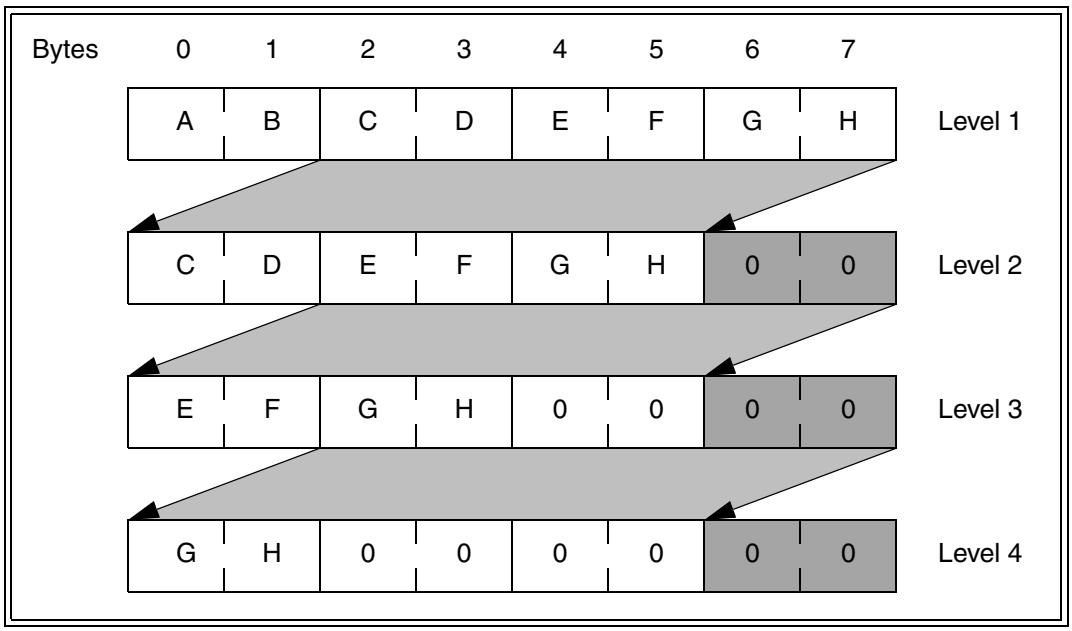


Figure 15 — Eight Byte logical unit number structure adjustments

Table 3 — 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 4.

Table 4 — Eight Byte logical unit number structure

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)							
1				FIRST LEVEL ADDRESSING				(LSB)
2	(MSB)							
3				SECOND LEVEL ADDRESSING				(LSB)
4	(MSB)							
5				THIRD LEVEL ADDRESSING				(LSB)
6	(MSB)							
7				FOURTH LEVEL ADDRESSING				(LSB)

The FIRST LEVEL ADDRESSING field specifies the first level address of a SCSI device. See table 5 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 5 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 5 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 5 for a definition of the FOURTH LEVEL ADDRESSING field.

The SCSI device pointed to in the FIRST LEVEL ADDRESSING, SECOND LEVEL ADDRESSING, THIRD LEVEL ADDRESSING, and FOURTH LEVEL ADDRESSING fields may be any physical or logical device addressable by an application client.

Table 5 — Format of addressing fields

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

The ADDRESS METHOD field defines the contents of the ADDRESS METHOD SPECIFIC field. See table 6 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 6 — ADDRESS METHOD field values

Code	Description	Reference
10b	Logical unit addressing method	4.9.5
00b	Peripheral device addressing method	4.9.6
01b	Flat space addressing method	4.9.7
11b	Extended logical unit addressing method	4.9.8

4.9.5 Logical unit addressing method

If the logical unit addressing method is selected the SCSI device should relay the received command 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.

NOTE 1 - A SCSI device may filter (i.e., not relay) commands to prevent commands 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).

See table 7 for the definition of the ADDRESS METHOD SPECIFIC field used when the logical unit addressing method is selected.

Table 7 — Logical unit addressing

Bit / Byte	7	6	5	4	3	2	1	0
n-1	1	0	TARGET					
n	BUS NUMBER			LUN				

The TARGET field, BUS NUMBER field, and LUN field address the logical unit to which the received command shall be relayed. The command 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 SCSI target device information in the TARGET field may be a target port identifier (see 4.7.2) or it may be a mapped representation of a target port identifier, when the range of possible target port identifiers is too large to fit in the TARGET field.

NOTE 2 - The value of target port identifiers within the TARGET field are defined by individual standards. (e.g., SCSI Parallel Interface -2 standard defines target port identifiers to be in the range 0 to 7, 0 to 15, and 0 to 31).

4.9.6 Peripheral device addressing method

If the peripheral device addressing method is selected the SCSI device should relay the received command 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.

NOTE 3 - A SCSI device may filter (i.e., not relay) commands to prevent commands 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).

See table 8 for the definition of the ADDRESS METHOD SPECIFIC field used when the peripheral device addressing method is selected.

Table 8 — Peripheral device addressing

Bit / Byte	7	6	5	4	3	2	1	0
n-1	0	0	BUS IDENTIFIER					
n	TARGET/LUN							

The BUS IDENTIFIER field identifies the bus or path that the SCSI device shall use to relay the received command. The BUS IDENTIFIER field may use the same value encoding as the BUS NUMBER field (see 4.9.5). However, bus identifier zero shall specify that the command is to be relayed to a logical unit within the SCSI device at the current level.

The TARGET/LUN field specifies the address of the peripheral device to which the SCSI device shall relay the received command. The meaning and usage of the TARGET/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 need LUNs but are not attached to SCSI buses (e.g., fans, cache, controllers, etc.). When the BUS IDENTIFIER field contains zero, the command shall be relayed to the current level logical unit specified by the TARGET/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 shall be relayed to the logical unit with the logical unit number zero within the SCSI target device specified in the TARGET/LUN field located in the SCSI domain specified by the BUS IDENTIFIER field. The SCSI target device information in the TARGET/LUN field may be a target port identifier (see 4.7.2) or it may be a mapped representation of a target port identifier, when the range of possible target port identifiers is too large to fit in the TARGET/LUN field.

NOTE 4 - The value of target port identifiers within the TARGET/LUN field are defined by individual standards. (e.g., SCSI Parallel Interface -2 standard defines target port identifiers to be in the range 0 to 7, 0 to 15, and 0 to 31).

The SCSI device located within the current level shall be addressed by a BUS IDENTIFIER field and a TARGET/LUN field of all zeros, also known as LUN 0 (see 4.9.2).

4.9.7 Flat space addressing method

All SCSI commands are allowed when the flat space addressing method is used, however, the addressed logical unit is not required to support all SCSI commands. Any command that is not supported 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.

In the response to an INQUIRY command, the addressed logical unit shall return a valid SCSI peripheral device type (e.g., direct access device, streaming device).

See table 9 for the definition of the ADDRESS METHOD SPECIFIC field used when the flat space addressing method is selected.

Table 9 — Flat space addressing

Bit / Byte	7	6	5	4	3	2	1	0
n-1	0	1	(MSB)					
n	LUN							(LSB)

The LUN field specifies the address of the logical unit to which the current level shall direct the received command.

4.9.8 Extended logical unit addressing

Extended logical unit addressing builds on the formats defined for dependent logical units (see 4.13) 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.

Extended logical units are identified by the ADDRESS METHOD field (see table 6 in 4.13) 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.

See table 10 for the definition of the ADDRESS METHOD SPECIFIC field used when the extended logical unit addressing method is selected.

Table 10 — Extended logical unit addressing

Bit / Byte	7	6	5	4	3	2	1	0
n	1	1	LENGTH		EXTENDED ADDRESS METHOD			
m	EXTENDED ADDRESS METHOD SPECIFIC							

The LENGTH field (see table 11) specifies the length of the EXTENDED ADDRESS METHOD SPECIFIC field.

Table 11 — LENGTH field values

Value	Length of the EXTENDED ADDRESS METHOD SPECIFIC Field	Reference
00b	One byte	table 12
01b	Three bytes	table 13
10b	Five bytes	table 14
11b	Seven bytes	table 15

Table 12, table 13, table 14, and table 15 show the four extended logical unit addressing formats.

Table 12 — Two byte extended logical unit addressing format

Bit / Byte	7	6	5	4	3	2	1	0
n	1	1	LENGTH (00b)		EXTENDED ADDRESS METHOD			
n+1	EXTENDED ADDRESS METHOD SPECIFIC							

Table 13 — Four byte extended logical unit addressing format

Bit / Byte	7	6	5	4	3	2	1	0
n	1	1	LENGTH (01b)		EXTENDED ADDRESS METHOD			
n+1	(MSB)							
	EXTENDED ADDRESS METHOD SPECIFIC							
n+3								(LSB)

Table 14 — Six byte extended logical unit addressing format

Bit / Byte	7	6	5	4	3	2	1	0
n	1	1	LENGTH (10b)		EXTENDED ADDRESS METHOD			
n+1	(MSB)							
	EXTENDED ADDRESS METHOD SPECIFIC							
n+5								(LSB)

Table 15 — Eight byte extended logical unit addressing format

Bit / Byte	7	6	5	4	3	2	1	0
0	1	1	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 16) specifies the type and size of extended logical unit address found in the EXTENDED ADDRESS METHOD SPECIFIC field.

Table 16 — Logical unit extended address methods

EXTENDED ADDRESS METHOD Code	**LENGTH Code(s)**	**Description**	**Reference**
0h	00b - 11b	Reserved	
1h	00b	Well known logical unit	4.9.9
1h	01b - 11b	Reserved	
2h - Fh	00b - 11b	Reserved	

4.9.9 Well known logical unit addressing

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

See table 17 for the definition of the EXTENDED ADDRESS METHOD SPECIFIC field used when the well known logical unit extended address method is selected.

Table 17 — Well known logical unit extended address format

Bit / Byte	7	6	5	4	3	2	1	0
n	1	1	LENGTH (00b)		Well known logical unit (1h)			
n+1	W-LUN							

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

4.10 Tasks

4.10.1 The task object

The task object represents either a tagged task or an untagged task. The composition of a task includes a definition of the work to be performed by the logical unit in the form of a command or a group of linked commands. A tagged task is represented by an I_T_L_Q nexus (see 4.11) and is composed of a definition of the work to be performed by the logical unit, and a task attribute (see 7.5). An untagged task is represented by an I_T_L nexus (see 4.11) and is composed of a definition of the work to be performed by the logical unit, and implicitly a SIMPLE task attribute (see 7.5).

The I_T_L_Q nexus representing a tagged task includes a tag (see 4.10.2) allowing many uniquely identified tagged tasks to be present concurrently in a single task set. A tagged task also includes one of the task attributes described in 7.5 that allows the application client to specify processing relationships between various tagged tasks. An untagged task does not include a tag in its I_T_L nexus, thus restricting the number of concurrent untagged tasks in a single task set to one per SCSI initiator port. An untagged task is assumed to have a SIMPLE task attribute.

Every SCSI transport protocol shall support tagged tasks and may support untagged tasks. If the SCSI transport protocol upon which a SCSI device operates supports untagged tasks, the SCSI device is not required to support tagged tasks.

An I_T_L_x 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 5.9.2). An I_T_L_x nexus is unique if one or more of its components is unique within the specified time interval. An untagged task shall be unique with respect to all tagged tasks in the task set.

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

4.10.2 Task tags

A tag is a field containing up to 64 bits that is a component of an I_T_L_Q nexus. A SCSI initiator device assigns tag values in each I_T_L_Q nexus in a way that ensures that the nexus uniqueness requirements stated in 4.10.1 are met.

4.11 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 tagged 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 18 maps the nexus object to other identifier objects.

Table 18 — Mapping nexus to SAM-2 identifiers

Nexus	Identifiers contained in nexus	Reference
I_T	Initiator Port Identifier Target Port Identifier	4.7.1 4.7.2
I_T_L	Initiator Port Identifier Target Port Identifier Logical Unit Number	4.7.1 4.7.2 4.8
I_T_L_Q	Initiator Port Identifier Target Port Identifier Logical Unit Number Tag	4.7.1 4.7.2 4.8 4.10.2

4.12 SCSI ports

4.12.1 SCSI port configurations

A SCSI device may contain only SCSI target ports, only SCSI initiator ports, only SCSI target/initiator ports or any combination of ports. Some of the port configurations possible for a SCSI device are shown in figure 16.

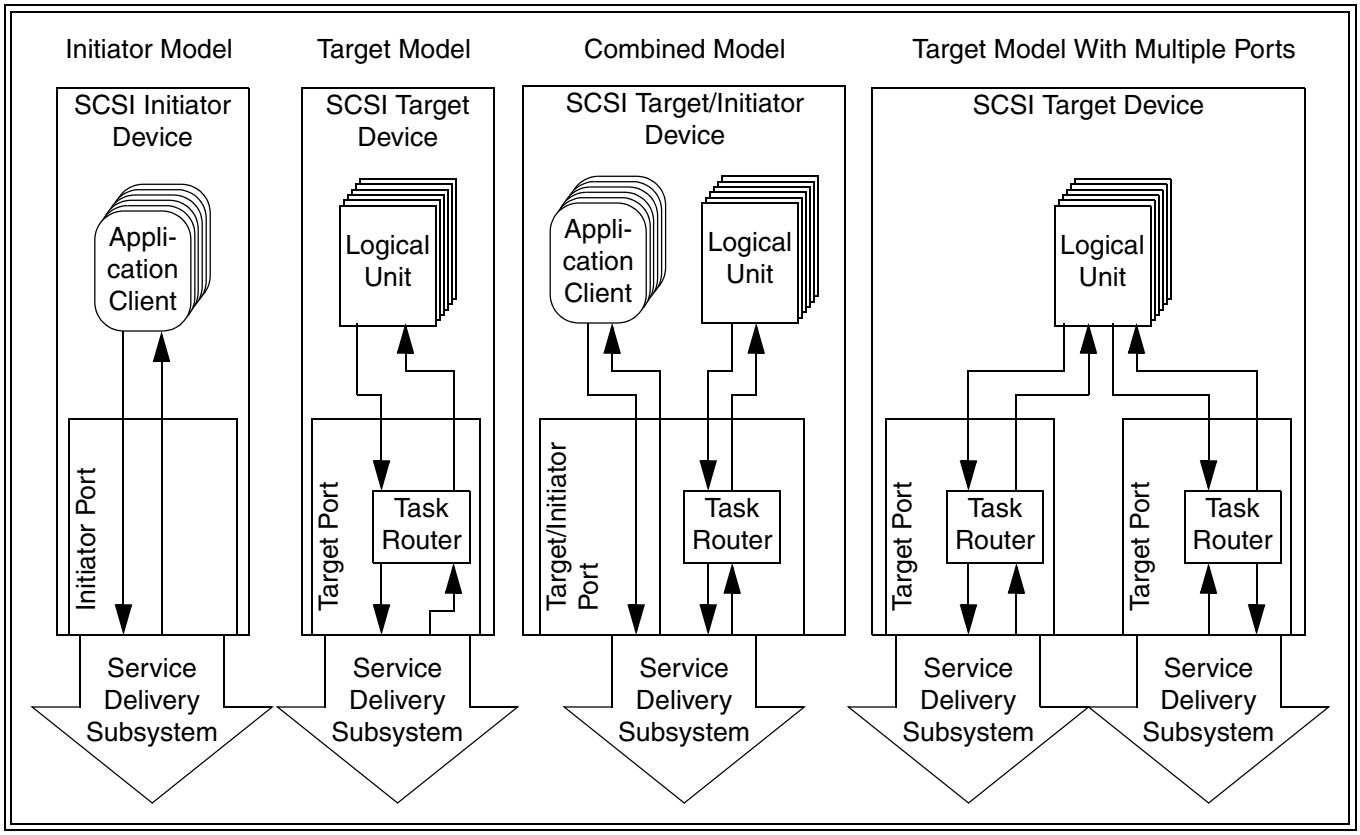


Figure 16 — SCSI device functional models

A SCSI target/initiator device is referred to by the role its port takes when it participates in an I/O operation. When a SCSI target/initiator device receives SCSI commands or task management functions, the SCSI target/initiator device takes on the characteristics of and is referred to as a SCSI target device. When a SCSI target/initiator device issues SCSI commands or task management functions, the SCSI target/initiator device takes on the characteristics of and is referred to as a SCSI initiator device.

4.12.2 SCSI devices with multiple ports

The model for a SCSI device with multiple ports is a single SCSI target device (see 4.7.2), SCSI initiator device (see 4.7.1), or SCSI target/initiator device (see 4.7.3) with multiple ports. Similarly, a single SCSI target port or SCSI initiator port may respond to multiple SCSI identifiers, with the model for such a SCSI port being one of multiple SCSI target ports or SCSI initiator ports (i.e., one for each SCSI identifier).

The SCSI identifiers representing the ports shall meet the requirements for initiator port identifiers (see 4.7.1) or target port identifiers (see 4.7.2) or both. SCSI target/initiator devices with multiple ports implement both target and initiator models and combine the SCSI target/initiator port structures in vendor specific ways that meet product requirements while maintaining the model for SCSI devices with multiple ports for the target and initiator functions

performed by the product. 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 SCSI target/initiator port, or a SCSI target port is servicing a SCSI initiator port or SCSI target/initiator port. The structures and views of SCSI devices are asymmetric for SCSI target ports and SCSI initiator ports.

4.12.3 Multiple port target SCSI device structure

Figure 17 shows the structure of a SCSI target device with multiple SCSI target ports. Each SCSI target port consists of a task router that is shared by a collection of logical units. Each logical unit contains a single task manager and a device server.

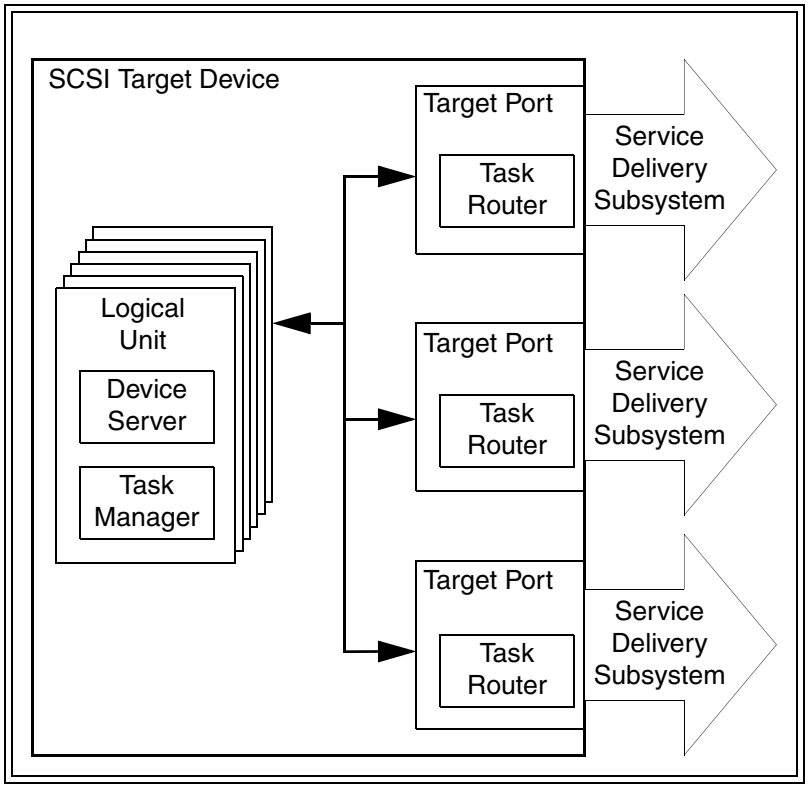


Figure 17 — Multiple port target SCSI device structure model

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 and the task router shall route them to a device server for processing. The REPORT LUNS commands (see SPC-2) shall be accepted by the logical unit with the logical unit number zero 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 SCSI port name or identifier values in the INQUIRY command Device Identification VPD page (see SPC-2).

4.12.4 Multiple port initiator SCSI device structure

Figure 18 shows the structure of a SCSI initiator device with multiple SCSI initiator ports. Each SCSI initiator port is shared by a collection of application clients.

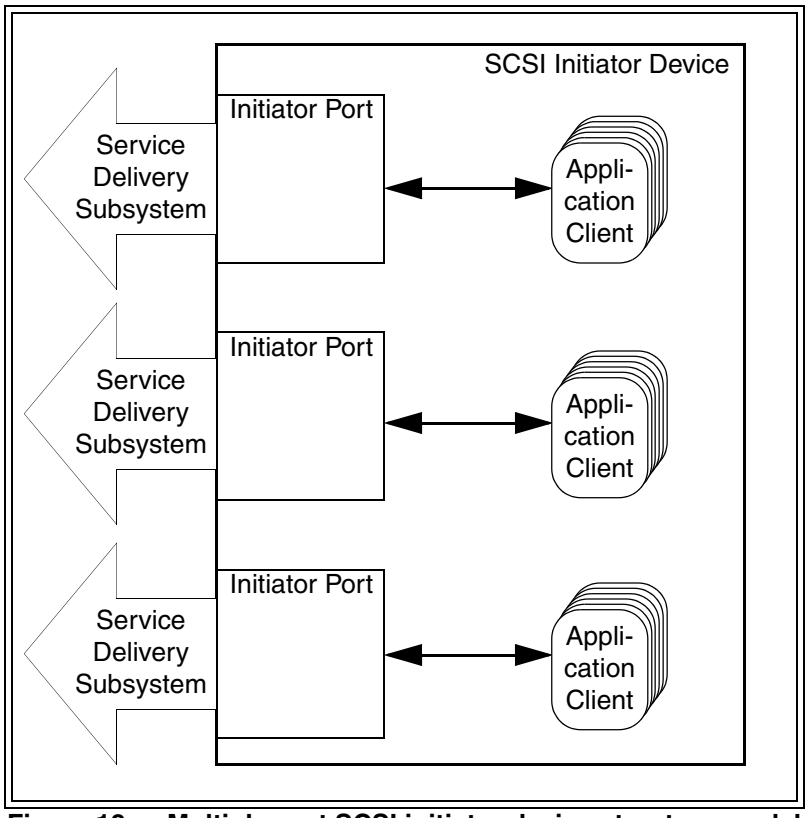


Figure 18 — Multiple port SCSI initiator device structure model

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).

4.12.5 Multiple port target/initiator SCSI device structure

Figure 19 shows the structure of a SCSI target/initiator device with multiple SCSI target/initiator ports. Each SCSI target/initiator port consists of a task router and is shared by a collection of logical units and application clients. Each logical unit contains a single task manager and a device server.

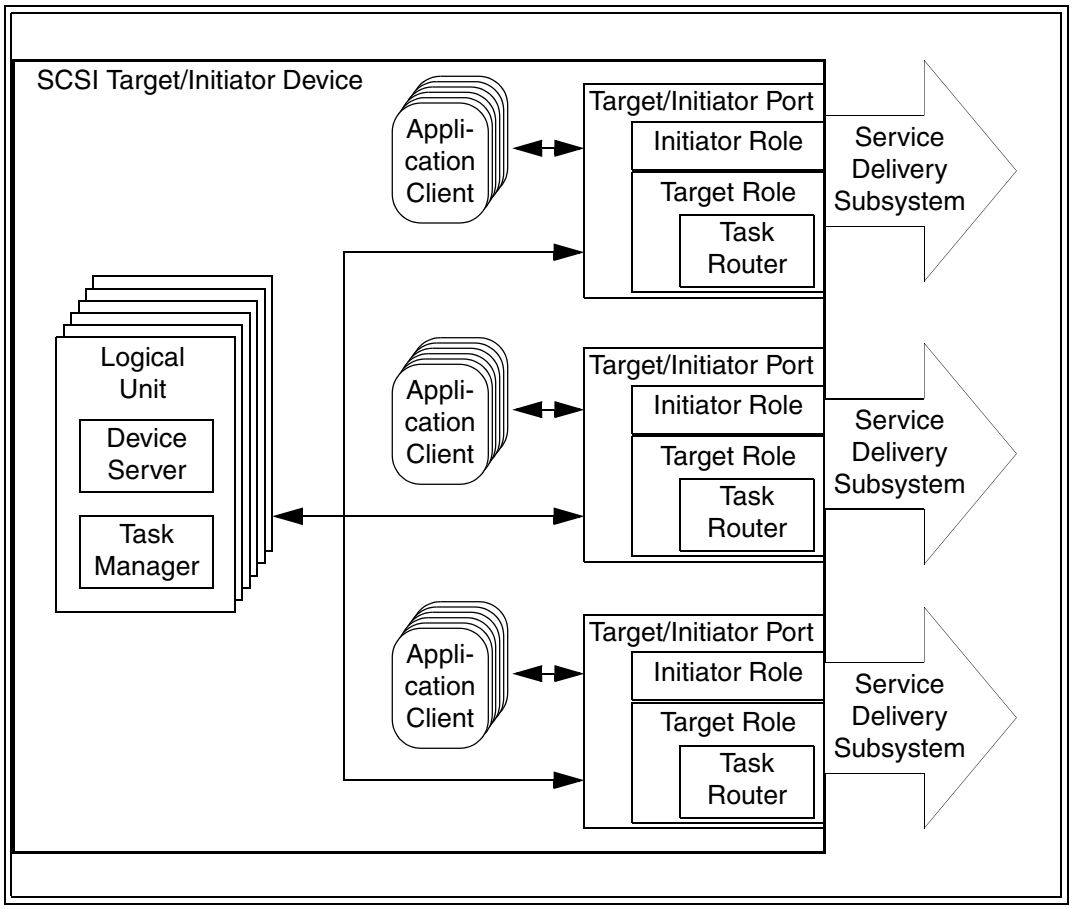


Figure 19 — Multiple port target/initiator SCSI device structure model

Two-way communications shall be possible between all logical units and all SCSI target/initiator ports, however, communications between any logical unit and any SCSI target/initiator port may be inactive. Two-way communications shall be possible between an application client and its associated SCSI target/initiator port. Each SCSI target/initiator port shall accept commands sent to LUN 0 and the task router shall route them to a device server for processing. The REPORT LUNS commands (see SPC-2) shall be accepted by the logical unit with the logical unit number zero from any SCSI target/initiator port and shall return the logical unit inventory available via that SCSI target/initiator port. The availability of the same logical unit through multiple SCSI target/initiator ports is discovered by matching SCSI port name or identifier values in the INQUIRY command Device Identification VPD page (see SPC-2).

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 initiator ports on a single SCSI target/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).

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

A SCSI target device may be connected to multiple domains such that a SCSI initiator port is only able to communicate with 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 20 and figure 21 show two examples of such configurations.

Figure 20 shows a SCSI target device with multiple SCSI target ports 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 SCSI target port identifiers and two initiator port identifiers in this SCSI domain. Using the INQUIRY command Device Identification VPD page (see SPC-2), the application clients in each of the SCSI initiator devices have the ability to discover the logical units in the SCSI target devices are accessible via multiple SCSI target port identifiers (i.e., SCSI target ports) and map the configuration of the SCSI target devices.

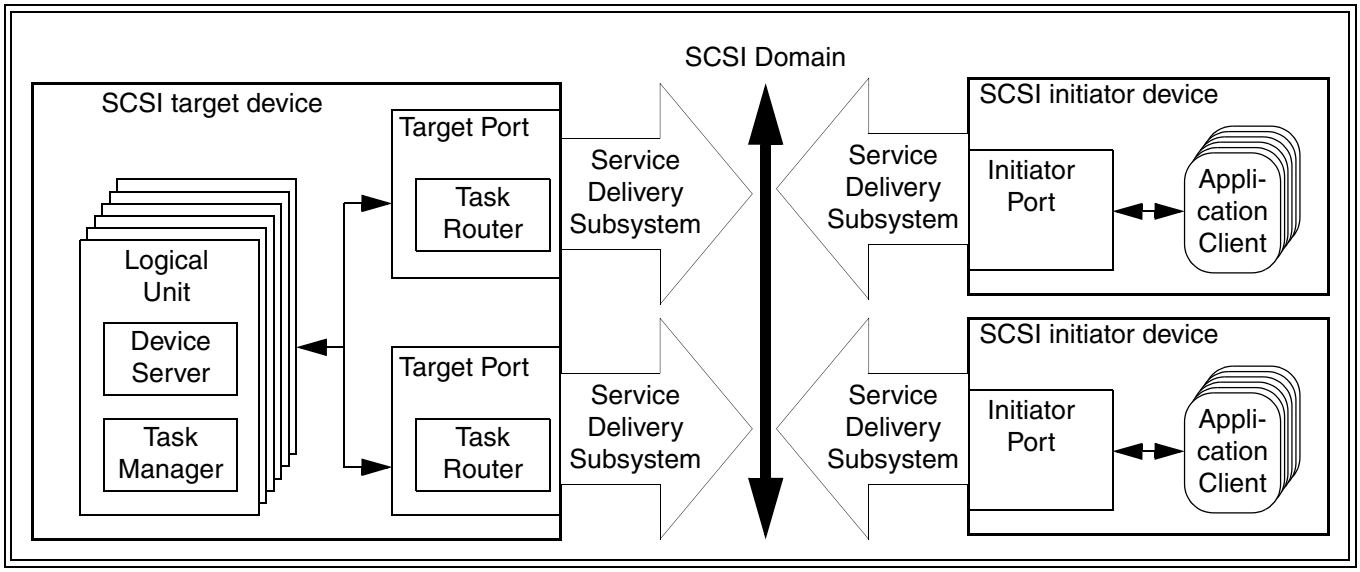


Figure 20 — SCSI target device configured in a single SCSI domain

Figure 21 shows a SCSI target device with multiple SCSI target ports participating in two SCSI domains and a SCSI initiator device with multiple SCSI initiator ports 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 SCSI 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-2), 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 ports and map the configuration. However, application clients may not be able to distinguish between the configuration shown in figure 21 and the configuration shown in figure 22.

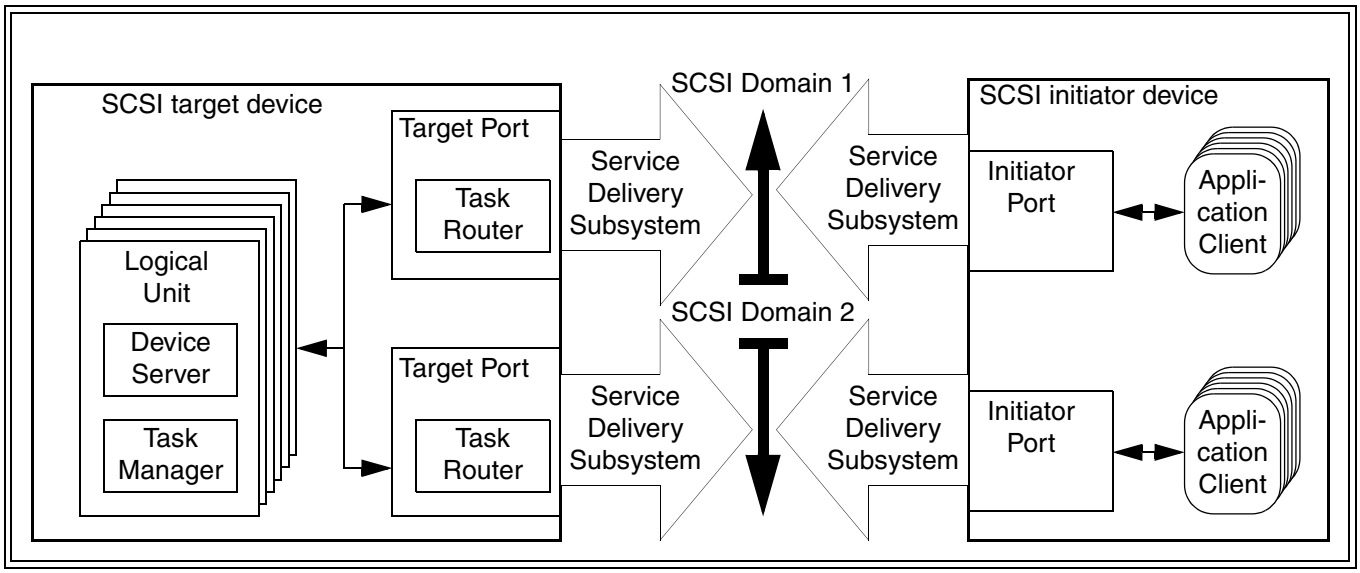


Figure 21 — SCSI target device configured in multiple SCSI domains

Figure 22 shows the same configuration as figure 21 except that the two SCSI domains have been replaced by a single SCSI domain.

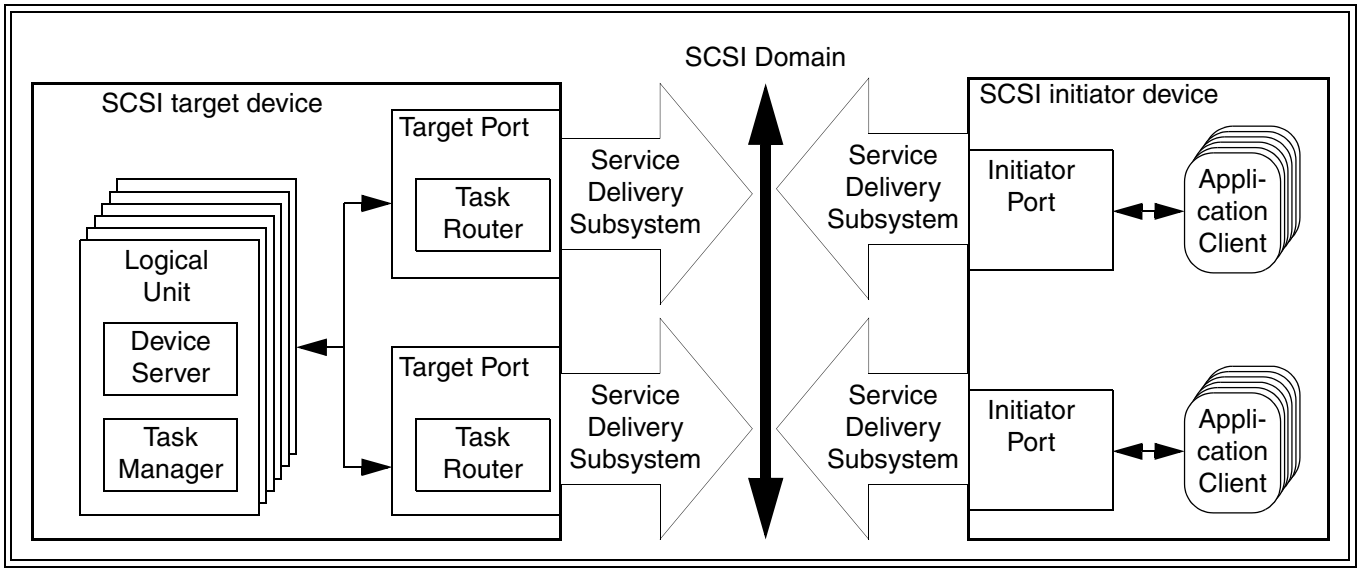


Figure 22 — SCSI target device and SCSI initiator device configured in a single SCSI domain

This model for application client determination of multiple SCSI target port configurations relies on information that is available only to the application clients via SCSI commands. The SCSI initiator ports in the SCSI initiator devices (figure 20) or SCSI initiator device (figure 21 and figure 22) are unable to distinguish the multiple SCSI target ports from individual SCSI target ports in two separate SCSI target devices.

4.12.7 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. For example, a SCSI target device handles the configurations shown in figure 21 and figure 22 in exactly the same way it handles the configuration show in figure 20.

NOTE 5 - The implications of this view of a SCSI initiator device are more far reaching than are immediately apparent. For example, if a SCSI initiator device makes an exclusive access reservation via one SCSI initiator port, then access will be denied to the other SCSI initiator port(s) on that same SCSI initiator device.

4.13 Model for dependent logical units

Optionally, the model for a logical unit (see 4.8) may include one or more unique logical units embedded within another logical unit. The embedded logical units are called dependent logical units (see 3.1.21). In such cases, the model hierarchy diagram in 4.8 is modified to become the diagram shown in figure 23.

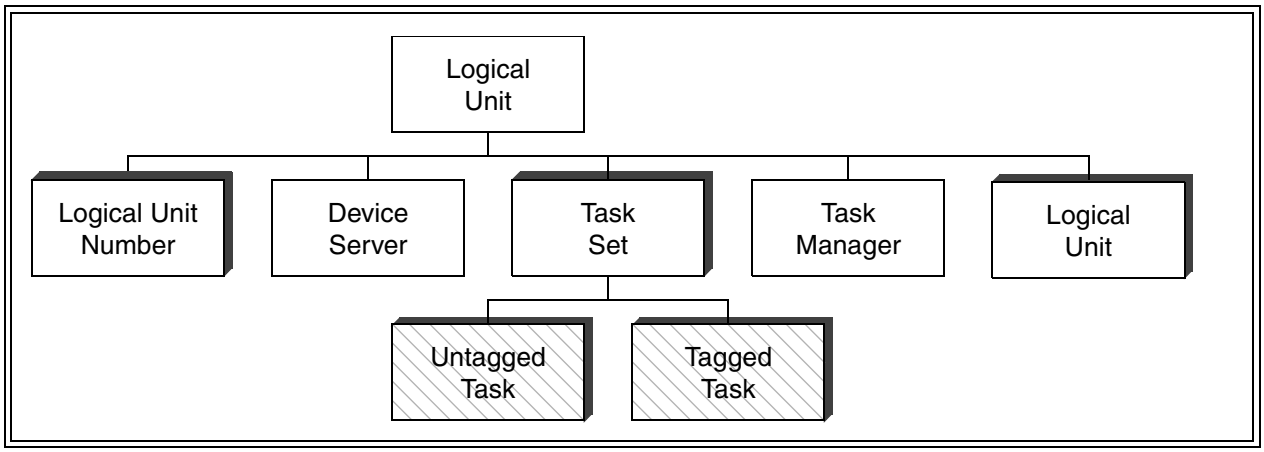


Figure 23 — Dependent logical unit model

When the dependent logical unit model is utilized, the hierarchical logical unit structure defined in 4.9.4 shall be used. If any logical unit within a SCSI target device includes dependent logical units, all logical unit numbers within the SCSI target device shall have the format described in 4.9.4. A device server that implements the hierarchical structure for dependent logical units described in this subclause shall set the HISUP bit to one in the standard INQUIRY data returned by the logical unit with the logical unit number zero (see SPC-2).

The hierarchical logical unit structure is an inverted tree containing up to four addressable levels. The example in figure 24 is a three-level system that consists of:

a) One SCSI initiator device that has three SCSI target devices attached in a single SCSI domain that is unable to add more SCSI target devices. One of the SCSI target devices is the level 1 SCSI device with dependent logical units (SDDLU). The level 1 SDDLU has two SCSI target ports, one in each of the SCSI domains containing a SCSI initiator device;
b) One SCSI initiator device has three SCSI target devices attached in a single SCSI domain that is able to add more SCSI target devices. One of the SCSI target devices is the level 1 SDDLU;
c) The level 1 SDDLU has three SCSI domains (called buses for hierarchical addressing purposes) with SCSI target devices attached and is capable of connecting more SCSI domains;
    A) Two of the SCSI domains contain two SCSI target devices each and these SCSI domains are unable to add more SCSI target devices. One of the SCSI target devices is the level 2 SDDLU;
    B) One of the SCSI domains contains two SCSI target devices and is able to add more SCSI target devices;
    C) The level 2 SDDLU has three SCSI domains with SCSI target devices attached and is capable of connecting more SCSI domains;
        a) Two of the SCSI domains contain two SCSI target devices each and these SCSI domains are unable to add more SCSI target devices; and
        b) One of the SCSI domains contains two SCSI target devices and is able to add more SCSI target devices.

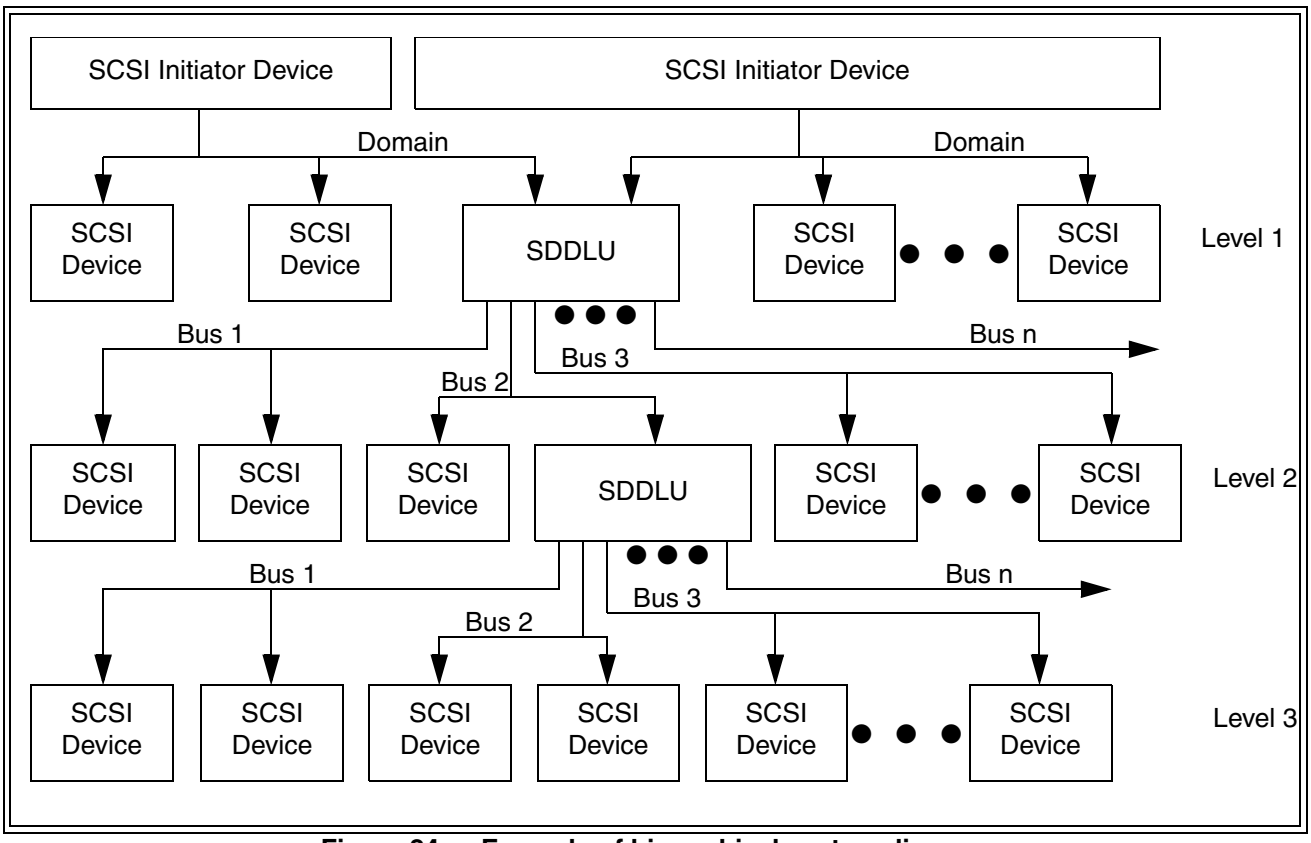


Figure 24 — Example of hierarchical system diagram

SCSI devices at each level in the tree are referenced by one of the following address methods:

a) Logical unit address method (see 4.9.5);
b) Peripheral device address method (see 4.9.6);
c) Flat space addressing method (see 4.9.7); or
d) Extended logical unit addressing method (see 4.9.8).

All peripheral device addresses, except LUN 0 (see 4.9.2), default to vendor specific values. All addressable entities 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 system there may be SCSI target devices that have multiple logical units connected to them through separate SCSI initiator ports. 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 4.9.5 and 4.9.6.

SCSI target devices shall assign a bus number of zero to all the logical units under control by the SCSI target device that are not connected through a SCSI domain where the SCSI target device functions as a SCSI initiator device.

4.14 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 25.

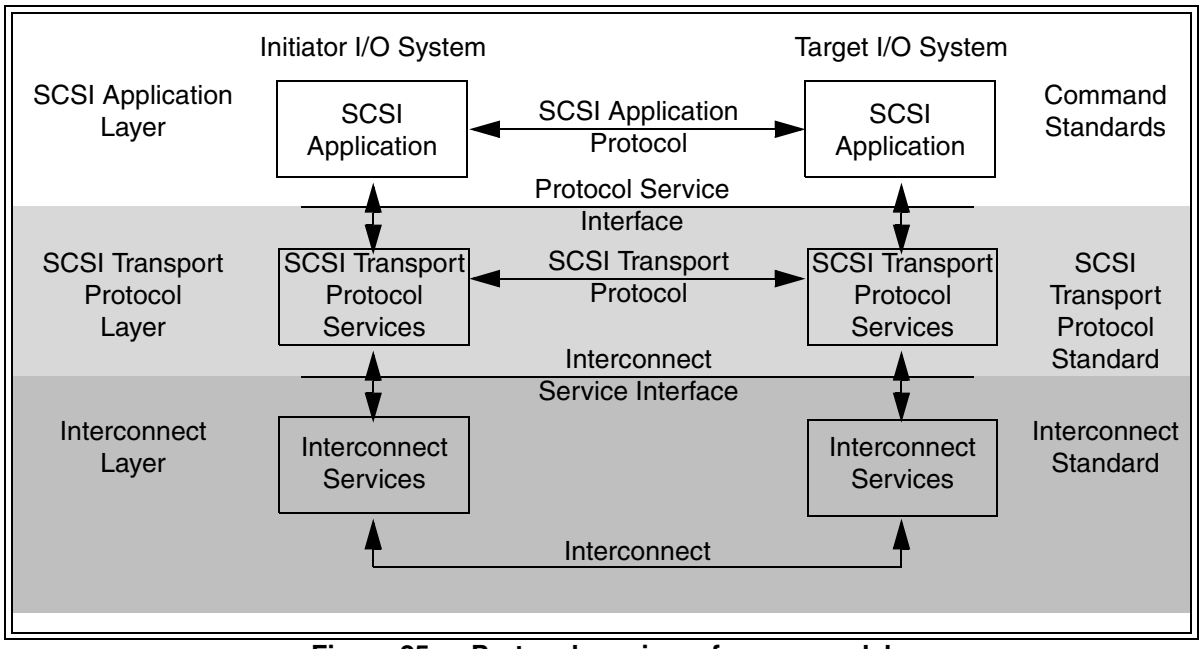


Figure 25 — Protocol service reference model

The layers comprising this model and the specifications defining the functionality of each layer are denoted by horizontal sequences. A layer consists of peer entities that communicate with one another by means of a protocol. Except for the interconnect layer, such communication is accomplished by invoking services provided by the adjacent lower layer. By convention, the layer from which a request for service originates is called the upper level protocol layer or ULP layer. The layer providing the service is referred to as the lower level protocol layer or LLP layer. The following layers are defined.

SCSI application layer: Contains the clients and servers that originate and process SCSI I/O operations by means of a SCSI application protocol;

SCSI transport protocol layer: Consists of the services and protocols through which clients and servers communicate; and

Interconnect layer: Comprised of the services, signaling mechanism and interconnect subsystem needed for the physical transfer of data from sender to receiver. In the SCSI model, the interconnect layer is known as the service delivery subsystem.

The set of protocol services implemented by the service delivery subsystem are intended to identify external behavioral requirements that apply to SCSI transport protocol standards. While these 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 protocol service interface is defined in this standard in representational terms using protocol services. The 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 ULP and LLP layers are defined with respect to the ULP layer and may originate in either layer. An outgoing interaction is modeled as a procedure call invoking an LLP service. An incoming interaction is modeled as a signal sent by the LLP layer that may be accompanied by parameters or data. Both types of interaction are described using the notation for procedures specified in 3.6.2. In this model, input arguments are defined relative to the layer receiving an interaction (i.e., an input is a parameter supplied to the receiving layer by the layer initiating the interaction).

The following types of service interactions between layers are defined:

SCSI transport protocol service request: A request from the ULP layer invoking a service provided by the LLP layer.

SCSI transport protocol service indication: A signal from the LLP layer informing the ULP layer that an asynchronous event has occurred (e.g., a reset or the receipt of a peer-to-peer protocol transaction).

SCSI transport protocol service response: A call to the LLP layer invoked by the ULP layer 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 ULP to the ULP peer.

SCSI transport protocol service confirmation: A signal from the LLP layer notifying the ULP layer 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 ULP peer.

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

Figure 26 shows the relationships between the four protocol service types.

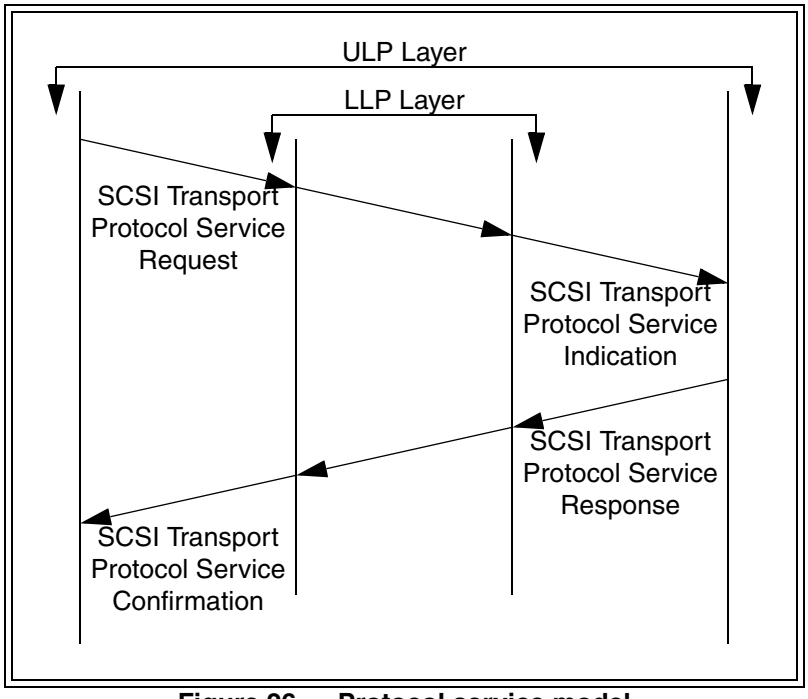


Figure 26 — Protocol service model

Figure 27 shows how protocol services may be used to process a client-server request-response transaction at the SCSI application layer.

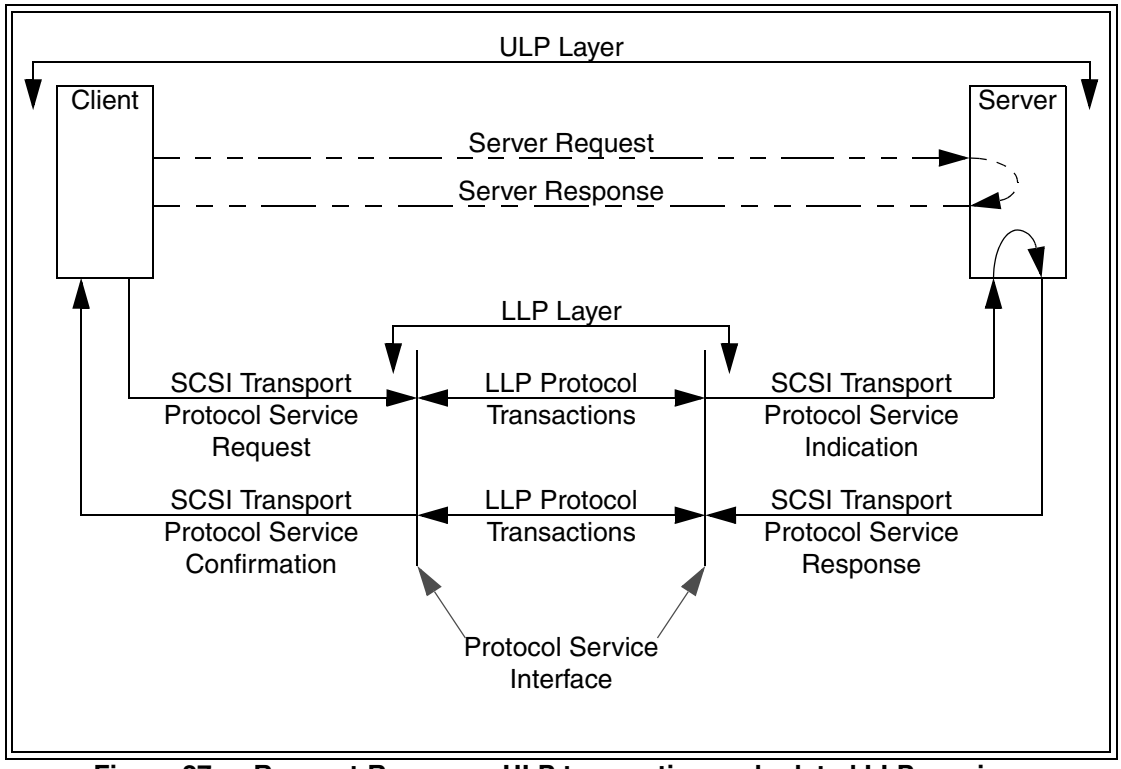


Figure 27 — Request-Response ULP transaction and related LLP services

The dashed lines in figure 27 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 27 show the corresponding SCSI transport protocol services and LLP transactions that are used to transport the data.

5 SCSI Command Model

5.1 The Execute Command remote procedure

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

Service response =Execute Command (IN (I_T_L_x Nexus, CDB, [Task Attribute], [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [Autosense Request], [Command Reference Number]), OUT ([Data-In Buffer], [Sense Data], Status))

Input Arguments:

I_T_L_x Nexus: Either an I_T_L nexus or an I_T_L_Q nexus (see 4.11).

CDB: Command descriptor block (see 5.2).

Task Attribute: A value specifying one of the task attributes defined in 7.5. This argument shall not be specified for an untagged command or the second and subsequent commands in a sequence of linked commands. Untagged tasks shall implicitly have the SIMPLE attribute. The attribute of a task that processes linked commands shall be set according to the Task Attribute argument specified for the first command in the sequence.

Data-In Buffer Size: The number of bytes available for data transfers to the Data-In Buffer (see 5.4.3).

Data-Out Buffer: A buffer containing command specific information to be sent to the logical unit, such as data or parameter lists needed to service the command. 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).

Autosense Request: An argument requesting the automatic return of sense data by means of the autosense mechanism specified in 5.9.4.3. It is not an error for the application client to provide this argument when autosense is not supported by the SCSI transport protocol or logical unit. SCSI transport protocols may require that the Autosense Request argument always request automatic return of the sense data.

Command Reference Number (CRN): When this argument is used, all sequential commands of an I_T_L nexus shall include a CRN argument that is incremented by one. The initial, wrap, and reset CRN values shall be one. The CRN value zero shall be reserved for use as defined by the SCSI transport protocol. It is not an error for the application client to provide this argument when CRN is not supported by the SCSI transport protocol or logical unit.

Output Arguments:

Data-In Buffer: A buffer to contain command specific information returned by the logical unit by the time of command completion. The application client shall not assume that the buffer contents are valid unless the command completes with a status of GOOD, CONDITION MET, INTERMEDIATE, or INTERMEDIATE-CONDITION MET. While some valid data may be present for other values of status, the application client should obtain additional information from the logical unit, such as sense data, to determine the state of the buffer contents. If the command ends with a service response of SERVICE DELIVERY OR TARGET FAILURE, the application client shall consider this parameter to be undefined.

Sense Data: A buffer to contain sense data returned by means of the autosense mechanism (see 5.9.4.3). If the command ends with a service response of SERVICE DELIVERY OR TARGET FAILURE, the application client shall consider this parameter to be undefined.

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

Service Response assumes one of the following values:

TASK COMPLETE: A logical unit response indicating that the task has ended. The status parameter shall have one of the values specified in 5.3 other than INTERMEDIATE or INTERMEDIATE-CONDITION MET.

LINKED COMMAND COMPLETE: Logical unit responses indicating that the task has not ended and that a linked command has completed successfully. As specified in 5.3, the status parameter shall have a value of INTERMEDIATE or INTERMEDIATE-CONDITION MET.

SERVICE DELIVERY OR TARGET FAILURE: The command has been ended due to a service delivery failure (see 3.1.108) or SCSI target device malfunction. All output parameters are invalid.

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

An application client requests processing of a linked command by setting the LINK bit to one in the CDB CONTROL byte as specified in 5.2.3. The task attribute is determined by the Task Attribute argument specified for the first command in the sequence. Upon receiving a response of LINKED COMMAND COMPLETE, an application client may issue the next command in the series through an **Execute Command** remote procedure call having the same I_T_L_x nexus and omitting the Task Attribute argument. If the logical unit receives the next command in a series of linked commands before completing the current command in that linked command series, the overlapped command condition described in 5.9.2 shall result.

5.2 Command descriptor block (CDB)

5.2.1 CDB format

The CDB defines the operation to be performed by the device server. For some commands, the CDB is accompanied by a list of command parameters contained in the Data-Out Buffer defined in 5.1. The parameters required for each command are specified in the applicable SCSI command standards.

If a logical unit validates reserved CDB fields and receives a reserved field within the CDB that is not zero or receives a reserved CDB code value, the logical unit shall terminate the command with CHECK CONDITION status; the sense key shall be set to ILLEGAL REQUEST with an additional sense code of INVALID FIELD IN CDB (see SPC-2).

For all commands, if the logical unit detects an invalid parameter in the CDB, then the logical unit shall complete the command without altering the media information.

All CDBs shall have an OPERATION CODE as the first byte. All CDBs, except the CDB for operation code 7Fh, shall have a CONTROL byte as the last byte. The format for the CDBs with operation code 7Fh is defined in SPC-2.

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.

The general format for all CDBs except the CDB for operation code 7Fh is shown in table 19. The remaining parameters depend on the command to be processed. All SCSI transport protocol standards shall accept CDBs less than or equal to 16 bytes in length. CDBs using the format shown in table 19 shall not exceed sixteen bytes in length.

Table 19 — Command Descriptor Block (CDB) Format

Bit Byte	7	6	5	4	3	2	1	0
0	OPERATION CODE							
1	Command specific parameters							
n-1								
n	CONTROL							

5.2.2 OPERATION CODE byte

The first byte of a SCSI CDB shall contain an operation code. The OPERATION CODE (see table 20) of the CDB has a GROUP CODE field and a COMMAND CODE field. The three-bit GROUP CODE field provides for eight groups of command codes. The five-bit COMMAND CODE field provides for thirty-two command codes in each group. A total of 256 possible operation codes exist. Operation codes are defined in the SCSI command standards. The group code value shall determine the length of the CDB (see table 21).

Table 20 — OPERATION CODE byte

Bit	7	6	5	4	3	2	1	0
	GROUP CODE			COMMAND CODE				

The value in the GROUP CODE field specifies one of the groups shown in table 21.

Table 21 — Group Code values

Group Code	Meaning
000b	6 byte commands
001b	10 byte commands
010b	10 byte commands
011b	reserved [a]
100b	16 byte commands
101b	12 byte commands
110b	vendor specific
111b	vendor specific

[a] The format the commands using the group code 011b and operation code 7Fh is described in SPC-2. With the exception of operation code 7Fh, all group code 011b operation codes are reserved.

5.2.3 CONTROL byte

The CONTROL byte is the last byte of every CDB, except for the CDB for operation code 7Fh. The CONTROL byte is defined in table 22.

Table 22 — CONTROL byte

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

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

The NACA (Normal ACA) bit is used to select whether a contingent allegiance (CA) or 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 a CA shall be established. The actions for CA and ACA are specified in 5.9.1.2. All logical units shall implement support for the NACA value of zero (i.e., CA) 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-2).

If the NACA bit is set to one but the logical unit does not support ACA, the logical unit shall complete the command with a CHECK CONDITION status, sense key of ILLEGAL REQUEST, an additional sense code of INVALID FIELD IN CDB and establish a CA condition. The requirements for handling the resulting CA condition shall be as described in 5.9.1.

The LINK bit is used to continue the task across multiple commands. Support for the LINK bit is optional. The application client sets the LINK bit to one to specify a request for continuation of the task across two or more SCSI commands. If the LINK bit is set to one and the command completes successfully, a logical unit that supports the LINK bit shall continue the task and return a status of INTERMEDIATE or INTERMEDIATE-CONDITION MET and a service response of LINKED COMMAND COMPLETE (see 5.3). The logical unit shall complete the command with a status of CHECK CONDITION and a sense key of ILLEGAL REQUEST if the LINK bit is set to one and the logical unit does not support linked commands.

Bit 1 provides an obsolete way to request interrupts between linked commands.

5.3 Status

5.3.1 Status codes

The status codes are specified in table 23. Status shall be sent from the device server to the application client whenever a command ends with a service response of TASK COMPLETE or LINKED COMMAND COMPLETE.

Table 23 — 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	INTERMEDIATE	No	LINKED COMMAND COMPLETE
14h	INTERMEDIATE-CONDITION MET	No	LINKED COMMAND COMPLETE
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 an CA or ACA condition has occurred (see 5.9.1). Autosense data may be delivered (see 5.9.4.3).

CONDITION MET. This status shall be returned whenever the requested operation specified by an unlinked command is satisfied (see the PRE-FETCH commands in the SBC 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 an unit attention condition to be established for the SCSI initiator port that sent the command with an additional sense code of PREVIOUS BUSY STATUS unless a PREVIOUS BUSY STATUS unit attention condition already exists.

INTERMEDIATE. This status or INTERMEDIATE-CONDITION MET shall be returned for each successfully completed command in a series of linked commands (except the last command), unless the command is terminated with CHECK CONDITION, RESERVATION CONFLICT, TASK SET FULL, or BUSY status. If INTERMEDIATE or INTERMEDIATE-CONDITION MET status is not returned, the series of linked commands is terminated and the task is ended. This status is the equivalent of GOOD status for linked commands.

INTERMEDIATE-CONDITION MET. This status is returned whenever the requested operation specified by a linked command is satisfied (see the PRE-FETCH commands in the SBC standard), unless the command is terminated with CHECK CONDITION, RESERVATION CONFLICT, TASK SET FULL, or BUSY status. If INTERMEDIATE or INTERMEDIATE-CONDITION MET status is not returned, the series of linked commands is terminated and the task is ended.

RESERVATION CONFLICT. This status shall be returned whenever a SCSI initiator port attempts to access a logical unit or an element of a logical unit in a way that conflicts with an existing reservation. (See the RESERVE, RELEASE, PERSISTENT RESERVE OUT and PERSISTENT RESERVE IN commands in SPC-2).

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 an unit attention condition to be established for the SCSI initiator port that sent the command with an additional sense code of PREVIOUS RESERVATION CONFLICT STATUS unless a PREVIOUS RESERVATION CONFLICT STATUS unit attention condition already exists.

TASK SET FULL. This status shall be implemented if the logical unit supports the creation of tagged tasks (see 4.10). This status shall not be implemented if the logical unit does not support the creation of tagged tasks.

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

When the logical unit has at least one task in the task set and a lack of task set resources prevents accepting a received untagged task into the task set, BUSY should be returned.

The logical unit should allow at least one command in the task set for each supported SCSI initiator port that has identified itself to the SCSI target port by a SCSI transport protocol specific procedure or by the successful transmission of a command.

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 an unit attention condition to be established for the SCSI initiator port that sent the command with an additional sense code of PREVIOUS TASK SET FULL STATUS unless a PREVIOUS TASK SET FULL STATUS unit attention condition already exists.

ACA ACTIVE. This status shall be returned when an ACA exists within a task set and a SCSI initiator port issues a command for that task set when at least one of the following is true:

a) There is a task with the ACA attribute (see 7.5.4) in the task set;
b) The SCSI initiator port issuing the command did not cause the ACA condition; or
c) The task created to process the command did not have the ACA attribute and the NACA bit was set to one in the CDB CONTROL byte of the faulting command (see 5.9.1).

The SCSI initiator port may reissue the command after the ACA condition has been cleared.

TASK ABORTED. This status shall be returned when a task is aborted by another SCSI initiator port and the Control mode page TAS bit is set to one (see 5.7.3).

5.3.2 Status precedence

If more than one condition applies to a completed task, the precedence for deciding the condition to be reported shall be:

1) Reporting a CHECK CONDITION status for any of the following unit attention conditions;
   a) POWER ON, RESET, OR BUS DEVICE RESET OCCURRED;
   b) POWER ON OCCURRED;
   c) SCSI BUS RESET OCCURRED;
   d) BUS DEVICE RESET FUNCTION OCCURRED;
   e) DEVICE INTERNAL RESET;
   f) TRANSCEIVER MODE CHANGED TO SINGLE-ENDED;
   g) TRANSCEIVER MODE CHANGED TO LVD; or
   h) I_T NEXUS LOSS OCCURRED;
2) Reporting a RESERVATION CONFLICT status;
3) Reporting a BUSY, RESERVATION CONFLICT, ACA ACTIVE or TASK SET FULL status; and
4) Reporting any other status.

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

5.4.2 Execute Command request/confirmation SCSI transport protocol services

All SCSI transport protocol standards shall define the SCSI transport protocol specific requirements for implementing the **Send SCSI Command** SCSI transport protocol service request and the **Command Complete Received** confirmation. Support for the **SCSI Command Received** indication and **Send Command Complete** response by a SCSI transport protocol standard is optional. All SCSI I/O systems shall implement these SCSI transport protocols as defined in the applicable SCSI transport protocol specification.

SCSI Transport Protocol Service Request:

Send SCSI Command (IN (I_T_L_x Nexus, CDB, [Task Attribute], [Data-In Buffer Size], [Data-Out Buffer], [Data-Out Buffer Size], [Autosense Request], [Command Reference Number]))

Input Arguments:

I_T_L_x Nexus: Either an I_T_L nexus or an I_T_L_Q nexus (see 4.11).

CDB: Command descriptor block (see 5.2).

Task Attribute: A value specifying one of the task attributes defined in 7.5. 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).

Data-Out Buffer: A buffer containing command specific information to be sent to the logical unit, such as data or parameter lists needed to service 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).

Autosense Request: An argument (see 5.1) requesting the automatic return of sense data by means of the autosense mechanism specified in 5.9.4.3.

Command Reference Number (CRN): When this argument is used, all sequential commands of an I_T_L nexus shall include a CRN argument that is incremented by one (see 5.1).

SCSI Transport Protocol Service Indication:

SCSI Command Received (IN (I_T_L_x Nexus, CDB, [Task Attribute], [Autosense Request], [Command Reference Number]))

Input Arguments:

I_T_L_x Nexus: Either an I_T_L nexus or an I_T_L_Q nexus (see 4.11).

CDB: Command descriptor block (see 5.2).

Task Attribute: A value specifying one of the task attributes defined in 7.5. For specific requirements on the Task Attribute argument see 5.1.

Autosense Request: This parameter is only present if the **Autosense Request** parameter was specified in the **Send SCSI Command** call and autosense delivery is supported by the SCSI transport protocol and logical unit.

Command Reference Number (CRN): When this argument is used, all sequential commands of an I_T_L nexus shall include a CRN argument that is incremented by one (see 5.1).

SCSI Transport Protocol Service Response (from device server)**:**

Send Command Complete (IN (I_T_L_x Nexus, [Sense Data], Status, Service Response))

Input Arguments:

I_T_L_x Nexus: Either an I_T_L nexus or an I_T_L_Q nexus (see 4.11).

Sense Data: If present, this argument instructs the SCSI target port to return sense information to the SCSI initiator port automatically (see 5.9.4.3).

Status: Command completion status (see 5.1).

Service Response: Possible service response information for the command (see 5.1).

SCSI Transport Protocol Service Confirmation:

Command Complete Received (IN (I_T_L_x Nexus, [Data-In Buffer], [Sense Data], Status, Service Response))

Input Arguments:

I_T_L_x Nexus: Either an I_T_L nexus or an I_T_L_Q nexus (see 4.11).

Data-In Buffer: A buffer containing command specific information returned by the logical unit on command completion (see 5.1).

Sense Data: Autosense data (see 5.9.4.3).

Status: Command completion status (see 5.1).

Service Response: Service response for the command (see 5.1).

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** remote 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 28). The model allows either unidirectional or bidirectional data transfer. The processing of a SCSI 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.

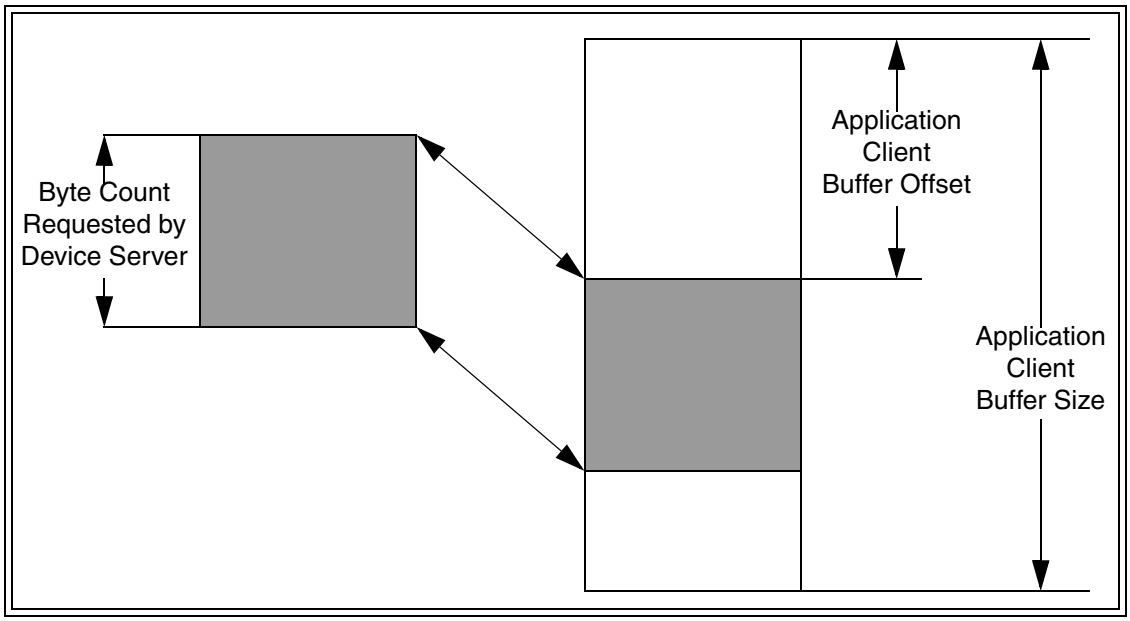


Figure 28 — Model for Data-In and Data-Out data transfers

It is assumed 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 SCSI 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 SCSI command. The amount of data moved per segment is usually a function of the buffering resources available to the logical unit. Figure 28 shows the model for such incremental data transfers.

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 (Data-In or Data-Out).

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 above arguments. A SCSI initiator device shall support a resolution of one byte. A SCSI target device may support any resolution.

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 monotonically 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 LLP confirmed services specified in 5.4.3.2 and 5.4.3.3 are used by the device server to request the transfer of command data to or from the application client. The SCSI initiator device SCSI transport protocol service interactions are unspecified.

The model 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

Request:

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

Argument descriptions:

I_T_L_x Nexus: either an I_T_L nexus or an I_T_L_Q nexus (see 4.11).

Device Server Buffer: Buffer to which data is to be transferred.

Application Client Buffer Offset: Offset in bytes from the beginning of the application client's buffer to the first byte of transferred data.

Request Byte Count: Number of bytes to be moved by this request.

Confirmation:

Data-In Delivered (IN (I_T_L_x Nexus))

This confirmation notifies the device server that the specified data was successfully delivered to the application client buffer.

Argument descriptions:

I_T_L_x Nexus: either an I_T_L nexus or an I_T_L_Q nexus (see 4.11).

5.4.3.3 Data-Out delivery service

Request:

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

Argument descriptions:

I_T_L_x Nexus: either an I_T_L nexus or an I_T_L_Q nexus (see 4.11).

Device Server Buffer: Buffer from which data is to be transferred.

Application Client Buffer Offset: Offset in bytes from the beginning of the application client's buffer to the first byte of transferred data.

Request Byte Count: Number of bytes to be moved by this request.

Confirmation:

Data-Out Received (IN (I_T_L_x Nexus))

This confirmation notifies the device server that the requested data has been successfully delivered to its buffer.

Argument descriptions:

I_T_L_x Nexus: either an I_T_L nexus or an I_T_L_Q nexus (see 4.11).

5.5 Task and command lifetimes

This subclause specifies the events delimiting the beginning and end (i.e., lifetime) of a task or pending SCSI 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 unless the command represents a continuation of a linked command as described in 5.1.

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 5.7.

The application client assumes that the task exists and 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 an unit attention condition with one of the following additional sense codes;
   A) COMMANDS CLEARED BY ANOTHER INITIATOR (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; TRANSCEIVER MODE CHANGED TO SINGLE-ENDED; or TRANSCEIVER MODE CHANGED TO LVD);

c) A service response of SERVICE DELIVERY OR TARGET FAILURE for the command. In this case, system implementations shall guarantee that the task associated with the failed command has ended;
d) A service response of FUNCTION COMPLETE following an ABORT TASK task management request 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 the specified task;
f) A service response of FUNCTION COMPLETE in response to a LOGICAL UNIT RESET task management function directed to the logical unit; or
g) A service response of FUNCTION COMPLETE following a TARGET RESET task management function directed to a SCSI target port with access to the logical unit.

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 above responses or a service response of LINKED COMMAND COMPLETE is received.

When a SCSI transport protocol does not require state synchronization (see 4.6.2), 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 normally is different from the lifetime observed by the device server.

5.6 Task management function lifetime

The application client assumes that the task management function is in process 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; or
b) Notification of an 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; TRANSCEIVER MODE CHANGED TO SINGLE-ENDED; or TRANSCEIVER MODE CHANGED TO LVD).

5.7 Aborting tasks

5.7.1 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) A logical unit reset (see 5.9.7);
c) A hard reset (see 5.9.6);
d) A power on condition; or
e) SCSI transport protocol specific events.

An action transmitted via a SCSI initiator port may abort task(s) created via the SCSI initiator port itself, task(s) created via another SCSI initiator port, or both its own tasks and tasks created via another SCSI initiator port.

The following actions affect only the task(s) created via the SCSI initiator port that transmits the action:

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 6.3;
c) An CA or ACA condition was established (see 5.9.1.2) and the QERR field was set to 01b or 11b in the Control mode page (see SPC-2); or
d) An ACA condition was cleared and the task had the ACA attribute (see 6.4).

The following actions affect the task(s) created via the SCSI initiator port that transmits the action and/or task(s) created via other SCSI initiator ports:

a) Completion of a CLEAR TASK SET task management function referencing the task set containing the specified task;
b) An CA or ACA condition was established (see 5.9.1.2) and the QERR field was set to 01b in the Control mode page (see SPC-2);
c) Completion of a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action with a reservation key that is associated with the SCSI initiator port that created the task (see SPC-2);
d) Completion of a LOGICAL UNIT RESET task management function (see 6.6) directed to the logical unit; or
e) Completion of a TARGET RESET task management function (see 6.8) directed to a SCSI target port with access to the logical unit.

5.7.2 When a SCSI initiator port aborts its own tasks

When a SCSI initiator port causes its own task(s) 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).

5.7.3 When a SCSI initiator port aborts tasks from other SCSI initiator ports

When a SCSI initiator port causes the task(s) of another SCSI initiator port to be aborted, the other SCSI initiator port 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-2) that applies to the other SCSI initiator port.

If the TAS bit is set to zero, the method of notification shall be an unit attention condition. The additional sense code set for the unit attention condition depends on the action that caused the task(s) to be aborted.

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. The COMMANDS CLEARED BY ANOTHER INITIATOR unit attention condition shall not be established, however, the establishment of any other applicable unit attention condition shall not be affected.

When a logical unit is aborting one or more tasks from a SCSI initiator port with the TASK ABORTED status it should complete all of those tasks before entering additional tasks from that SCSI initiator port into the task set.

5.8 Command processing examples

5.8.1 Unlinked command example

An unlinked command is used to show the events associated with the processing of a single device service request (see figure 29). This example does not include error or exception conditions.

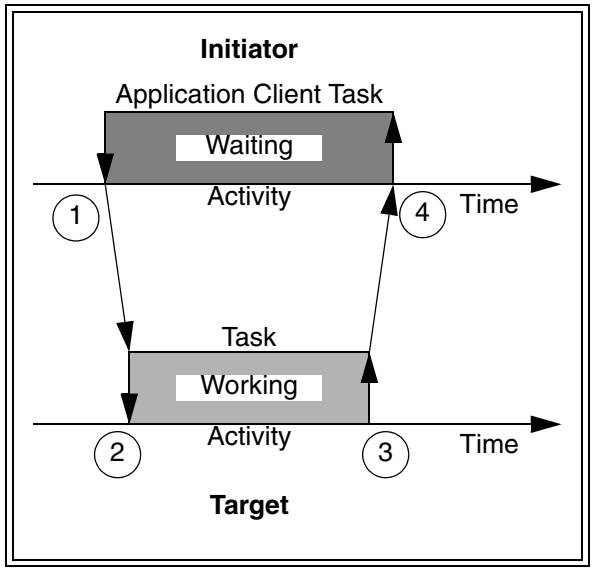


Figure 29 — Command processing events

The numbers in figure 29 identify the events described as follows:

1) The application client task performs an **Execute Command** remote 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.

5.8.2 Linked command example

A task may consist of multiple commands linked together. After the logical unit notifies the application client that a linked command has successfully completed, the application client issues the next command in the series.

The example in figure 30 shows the events in a sequence of two linked commands.

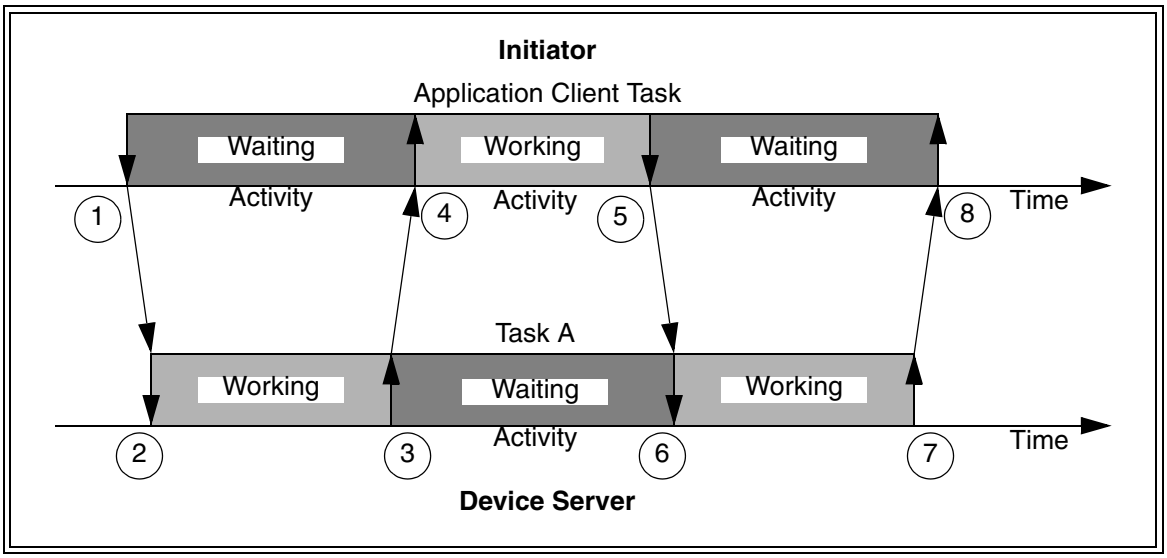


Figure 30 — Linked command processing events

The numbers in figure 30 Identify the events described as follows:

1) The application client task performs an **Execute Command** remote procedure call by invoking the **Send SCSI Command** SCSI transport protocol service to send the CDB and other input parameters to the logical unit. The LINK bit is set to one in the CDB CONTROL byte (see 5.2.3).

2) The device server is notified through a **SCSI Command Received** indication containing the CDB and command parameters. A task (Task A) is created and entered into the task set.

3) Upon completion of the first command, the device server invokes the **Send Command Complete** SCSI transport protocol service with the status set to INTERMEDIATE or INTERMEDIATE-CONDITION MET and a service response of LINKED COMMAND COMPLETE. Task A is not terminated.

4) The SCSI initiator port returns the status and service response to the application client task by means of a **Command Complete Received** confirmation.

5) The application client task performs an **Execute Command** remote procedure call by means of the **Send SCSI Command** SCSI transport protocol service as described in 1). The Task Attribute argument is omitted. The LINK bit in the CDB CONTROL byte is set to zero.

6) The device server receives the last command in the sequence and processes the operation.

7) The command completes successfully. Task A is terminated. A **Send Command Complete** SCSI transport protocol service response of TASK COMPLETE, with status GOOD, is sent to the application client.

8) The SCSI initiator port delivers an **Command Complete Received** confirmation containing the service response and status to the application client task.

5.9 Command processing considerations and exception conditions

5.9.1 Contingent allegiance (CA) and auto contingent allegiance (ACA)

5.9.1.1 Overview

There are two mechanisms for returning sense data when a command is terminated with a CHECK CONDITION status: autosense (see 5.9.4.3) and the REQUEST SENSE command (see SPC-2). There are two mechanisms for altering task processing when a command is terminated with a CHECK CONDITION status: CA and ACA. CA alters task processing so that sense data is preserved for subsequent delivery. ACA alters task processing until a CLEAR ACA task management function (see 6.4) is requested. Table 24 provides an overview of how autosense, CA, and ACA interact.

Table 24 — Autosense, CA, and ACA Interactions

Autosense Requested [a]	NACA Value [b]	Tasks Blocked [c]	
		From	To [d]
No	0 (i.e., CA)	Termination of a command with CHECK CONDITION status	Receipt of a command [e]
	1 (i.e., ACA)		Receipt of CLEAR ACA [g]
Yes	0 (i.e., CA)		Transmission of autosense data [f]
	1 (i.e., ACA)		Receipt of CLEAR ACA [g]

[a] Autosense is requested via the **Execute Command** remote procedure call (see 5.1).

[b] The NACA bit is in the CONTROL byte in the CDB (see 5.2.3).

[c] The blocking of tasks is described in 5.9.1.2. If the QERR field in the Control mode page (see SPC-2) contains 01b or 11b, tasks are aborted instead of being blocked. If the TST field in the Control mode page contains 000b, tasks from all SCSI initiator ports are blocked or aborted. If the TST field in the Control mode page contains 001b, only tasks from the faulted initiator port are blocked or aborted.

[d] This table covers only the normal methods for clearing a CA or ACA as seen by the faulted initiator port. Exception handling methods for clearing CA and ACA are described in 5.9.1.6 and 5.9.1.7.

[e] The intent is that the next command from the faulted initiator port be a REQUEST SENSE command but the next command received clears the CA condition, regardless of what command that is.

[f] Since the autosense data is transmitted coincident with the delivery of the CHECK CONDITION status (see 5.9.4.3), the interval during which tasks are blocked is not detectable by the application client. If the QERR field in the Control mode page (see SPC-2) contains 01b or 11b, the specified blocked tasks are aborted, an action that makes the CA condition detectable by the application client.

[g] The CLEAR ACA task management function is described in 6.4. During ACA new tasks received by the logical unit are not allowed to enter the task set unless they have the ACA task attribute (see 7.5.4) and are from the faulted initiator port. One of the results of the ACA task attribute requirement is that commands in-flight when the CHECK CONDITION status occurs are returned unprocessed to the SCSI initiator port with an ACA ACTIVE status. Multiple commands may be sent one at a time using the ACA task attribute to recover from the CHECK CONDITION that caused the ACA condition without clearing the ACA.

5.9.1.2 Establishing a CA or ACA

When a device server terminates a command with a CHECK CONDITION status, either an CA or ACA condition is established within the task set. If the NACA bit was zero in the CONTROL byte (see 5.2.3) of the faulting command, the device server shall create a CA condition. If the NACA bit was one in the CONTROL byte of the faulting command, the device server shall create an ACA condition.

When a CA or ACA condition is established, tasks in the dormant or enabled task state (see 7.4) shall either be aborted or blocked based on the contents of the TST and QERR field in the Control mode page (see SPC-2) as shown in table 25. The TST (task set type) Control mode page field specifies the type of task set in the logical unit (see SPC-2). The QERR (queue error management) Control mode page field specifies how the device server handles blocked and dormant tasks when another task receives a CHECK CONDITION status (see SPC-2).

Table 25 — Blocking and aborting tasks when a CA or ACA is established

QERR	TST	Action
00b	000b	All enabled tasks from all SCSI initiator ports shall transition to the blocked task state (see 7.6). All dormant tasks from all SCSI initiator ports shall remain in the dormant task state.
	001b	All enabled tasks from the faulted initiator port shall transition to the blocked task state (see 7.6). All dormant tasks from the faulted initiator port shall remain in the dormant task state. All tasks from SCSI initiator ports other than the faulted initiator port shall not be affected by the establishment of this CA or ACA condition.
01b	000b	All enabled and dormant tasks from all SCSI initiator ports shall be aborted (see 5.7).
	001b	All enabled and dormant tasks from the faulted initiator port shall be aborted (see 5.7). All tasks from SCSI initiator ports other than the faulted initiator port shall not be affected by the establishment of this CA or ACA condition.
11b	000b	All enabled and dormant tasks from the faulted initiator port shall be aborted (see 5.7). All enabled tasks from SCSI initiator ports other than the faulted initiator port shall transition to the blocked task state (see 7.6). All dormant tasks from SCSI initiator ports other than the faulted initiator port shall remain in the dormant task state.
	001b	All enabled and dormant tasks from the faulted initiator port shall be aborted (see 5.7). All tasks from SCSI initiator ports other than the faulted initiator port shall not be affected by the establishment of this CA or ACA condition.

After the CA or ACA condition is established:

a) New tasks from the faulted initiator port shall be handled as described in 5.9.1.4, and
b) New tasks from SCSI initiator ports other than the faulted initiator port shall be handled as described in 5.9.1.5.

A CA or ACA condition shall not cross task set boundaries and shall be preserved until it is cleared as described in 5.9.1.6 or 5.9.1.7. If requested by the application client and supported by the SCSI transport protocol and logical unit, sense data shall be returned via autosense as described in 5.9.4.3.

If the SCSI transport protocol does not enforce state synchronization as described in 4.6.2, there may be a time delay between the occurrence of the CA or ACA condition and the time at which the application client becomes aware of the condition.

5.9.1.3 Handling tasks when neither CA or ACA is in effect

Table 26 describes the handling of tasks when neither a CA nor an ACA condition is in effect for the task set. The number of SCSI initiator ports in the task set is influenced by the TST field in the Control mode page (see SPC-2).

Table 26 — Task handling when neither CA nor ACA is in effect

New Task Properties		Device Server Action	Condition Established if New Task Terminates with a CHECK CONDITION status
Attribute [a]	NACA Value [b]		
Any Attribute Except ACA	0	Process the task. [c]	CA
	1		ACA
ACA	0	Terminate the command with CHECK CONDITION status, sense key of ILLEGAL REQUEST and additional sense code of INVALID MESSAGE ERROR.	CA
	1		ACA

[a] Task attributes are described in 7.5.
[b] The NACA bit is in the CONTROL byte in the CDB (see 5.2.3).
[c] All the conditions that affect the processing of commands (e.g., reservations) still apply.

5.9.1.4 Handling new tasks from the faulted initiator port when CA or ACA is in effect

Table 27 describes the handling of new tasks from the faulted initiator port when CA is in effect.

Table 27 — Handling for new tasks from a faulted initiator port during CA

New Task Properties		Device Server Action	Condition Established If New Task Terminates with a CHECK CONDITION status [c]
Attribute [a]	NACA Value [b]		
Any Attribute Except ACA	0	Process the task. [d]	CA
	1		ACA
ACA	0	Terminate the command with CHECK CONDITION status, sense key of ILLEGAL REQUEST, and additional sense code of INVALID MESSAGE ERROR.	CA
	1		ACA

[a] Task attributes are described in 7.5.
[b] The NACA bit is in the CONTROL byte in the CDB (see 5.2.3).
[c] The CA condition is cleared upon completion of any new task regardless of status. Termination of that new task with CHECK CONDITION status shall result in the establishment of a new CA or ACA based on the value of the NACA bit.
[d] All the conditions that affect the processing of commands (e.g., reservations) still apply.

Table 28 describes the handling of new tasks from the faulted initiator port when ACA is in effect.

Table 28 — Handling for new tasks from a faulted initiator port during ACA

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

[a] Task attributes are described in 7.5.

[b] The NACA bit is in the CONTROL byte in the CDB (see 5.2.3).

[c] If a task with the ACA attribute terminates with a CHECK CONDITION status, the existing ACA condition shall be cleared and a new CA or ACA condition shall be established based on the value of the NACA bit.

[d] All the conditions that affect the processing of commands (e.g., reservations) still apply.

5.9.1.5 Handling new tasks from non-faulted initiator ports when CA or ACA is in effect

5.9.1.5.1 Commands permitted from non-faulted initiator ports during CA or ACA

The device server shall process a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action (see SPC-2) while a CA or ACA condition is established when the command is received from a SCSI initiator port other than the faulted initiator port.

NOTE 6 - The processing of specific commands (e.g., PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action) the from SCSI initiator ports other than the faulted initiator port while a CA or ACA condition is in effect provides SCSI initiator ports other than the faulted initiator port the opportunity to recover from error conditions that the faulted initiator port cannot recover from itself.

5.9.1.5.2 Handling new tasks from non-faulted initiator ports when CA or ACA is in effect

The handling of tasks created by SCSI initiator ports other than the faulted initiator port depends on the value in the TST field in the Control mode page (see SPC-2).

Table 29 describes the handling of new tasks from SCSI initiator ports other than the faulted initiator port when CA is in effect.

Table 29 — Handling for new tasks from non-faulted initiator ports during CA

TST Field Value in Control mode page	New Task Properties		New Command Permitted During CA [c]	Device Server Action	Condition 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 BUSY status.	n/a
	Any Attribute Except ACA	0 or 1	No	Terminate the task with BUSY status.	n/a
		0	Yes	Process the task.	CA [d]
		1	Yes		ACA [d]
001b	ACA	0	n/a	Terminate the command with CHECK CONDITION status, sense key of ILLEGAL REQUEST and additional sense code of INVALID MESSAGE ERROR.	CA
		1			ACA
	Any Attribute Except ACA	0 or 1	n/a	Process the task. [e]	See 5.9.1.3.

[a] Task attributes are described in 7.5.

[b] The NACA bit is in the CONTROL byte in the CDB (see 5.2.3).

[c] See 5.9.1.5.1.

[d] If a permitted command terminates with a CHECK CONDITION status, the existing CA condition shall be cleared and a new CA or ACA condition shall be established for a new faulted initiator port based on the value of the NACA bit.

[e] When the TST field in the Control mode page contains 001b, commands from SCSI initiator ports other than the faulted initiator port shall be processed as if the CA condition does not exist (see 5.9.1.3). In this case, the logical unit shall be capable of handling concurrent CA conditions and sense data for all SCSI initiator ports.

Table 30 describes the handling of new tasks from SCSI initiator ports other than the faulted initiator port when ACA is in effect.

Table 30 — Handling for new tasks from non-faulted initiator ports during ACA

TST Field Value in Control mode page	New Task Properties		New Command Permitted During ACA [c]	Device Server Action	Condition 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.	CA [d]
		1	Yes		ACA [d]
001b	ACA	0	n/a	Terminate the command with CHECK CONDITION status, sense key of ILLEGAL REQUEST and additional sense code of INVALID MESSAGE ERROR.	CA
		1			ACA
	Any Attribute Except ACA	0 or 1	n/a	Process the task. [e]	See 5.9.1.3.

[a] Task attributes are described in 7.5.

[b] The NACA bit is in the CONTROL byte in the CDB (see 5.2.3).

[c] See 5.9.1.5.1.

[d] If a permitted command terminates with a CHECK CONDITION status, the existing ACA condition shall be cleared and a new CA or ACA condition shall be established for a new faulted initiator port based on the value of the NACA bit.

[e] When the TST field in the Control mode page contains 001b, commands from SCSI initiator ports other than the faulted initiator port shall be processed as if the ACA condition does not exist (see 5.9.1.3). In this case, the logical unit shall be capable of handling concurrent ACA conditions and sense data for all SCSI initiator ports.

5.9.1.6 Clearing a CA condition

A CA condition shall only be cleared:

a) As a result of a power on or logical unit reset (see 5.9.7);
b) By an ABORT TASK SET task management function (see 6.3) from the faulted initiator port;
c) By a CLEAR TASK SET task management function (see 6.5) from any SCSI initiator port including the faulted initiator port if the TST field in the Control mode page (see SPC-2) contains 000b;
d) By a CLEAR TASK SET task management function from the faulted initiator port if the TST field in the Control mode page (see SPC-2) contains 001b;
e) By a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action from another SCSI initiator port that clears the tasks of the faulted initiator port (see SPC-2);
f) When a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action from another SCSI initiator port terminates in a CHECK CONDITION status;
g) Upon completion of a subsequent REQUEST SENSE command for the I_T_L nexus;
h) Upon accepting any subsequent command other than a REQUEST SENSE command for the I_T_L nexus; or
i) Upon sending sense data by means of the autosense mechanism (see 5.9.4.3).

Case f) results in the establishment of a new CA or ACA for a new faulted initiator port based on the value of the NACA bit.

When a CA condition is cleared and no new CA or ACA condition is established, the state of all tasks in the task set shall be modified as described in clause 7.

5.9.1.7 Clearing an ACA condition

An ACA condition shall only be cleared:

a) As the result of a power on or a logical unit reset (see 5.9.7);
b) By a CLEAR ACA task management function (see 6.4) from the faulted initiator port;
c) By a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action with the ACA task attribute from the faulted initiator port that clears the tasks of the faulted initiator port (see SPC-2);
d) By a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action with a task attribute other than ACA from a SCSI initiator port other than the faulted initiator port that clears the tasks of the faulted initiator port;
e) When a command with the ACA task attribute from the faulted initiator port 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) result in the establishment of a new CA or ACA based on the value of the NACA bit.

When an ACA condition is cleared and no new CA or ACA condition is established, the state of all tasks in the task set shall be modified as described in clause 7.

5.9.2 Overlapped commands

An overlapped command occurs when a task manager detects the use of a duplicate I_T_L_x nexus (see 4.10.1) in a command before a pending task holding that I_T_L_x nexus completes its task lifetime (see 5.5). Each SCSI transport protocol standard shall specify whether or not a task manager is required to detect overlapped commands.

A task manager that detects an overlapped command shall abort all tasks for the faulted initiator port in the task set and the device server shall return CHECK CONDITION status for that command. The sense key shall be set to ABORTED COMMAND and the additional sense code shall be set to OVERLAPPED COMMANDS ATTEMPTED.

NOTES

7 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).

8 Some logical units may not detect an overlapped command until after the CDB has been received.

5.9.3 Incorrect logical unit selection

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

The logical unit number may be incorrect because:

a) The SCSI target device does not support the logical unit (e.g., some SCSI target devices support only one peripheral device).

   In response to any other command except REQUEST SENSE and INQUIRY, the SCSI target device shall terminate the command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to LOGICAL UNIT NOT SUPPORTED;

b) The SCSI target device supports the logical unit, but the peripheral device is not currently attached to the SCSI target device.

   In response to an INQUIRY command the SCSI target device shall return the INQUIRY data with the peripheral qualifier set to the value required in SPC-2. In response to a REQUEST SENSE command, the SCSI target device shall return sense data. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to LOGICAL UNIT NOT SUPPORTED.

   In response to any other command except REQUEST SENSE and INQUIRY, the SCSI target device shall terminate the command with CHECK CONDITION status. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to LOGICAL UNIT NOT SUPPORTED;

c) The SCSI target device supports the logical unit and the peripheral device is attached, but not operational.

   In response to an INQUIRY command the SCSI target device shall return the INQUIRY data with the peripheral qualifier set to the value required in SPC-2. In response to REQUEST SENSE, the SCSI target device shall return sense data appropriate to the condition that is making the logical unit not operational.

   The SCSI target device's response to any command other than INQUIRY and REQUEST SENSE is vendor specific; or

d) The SCSI target device supports the logical unit but is incapable of determining if the peripheral device is attached or is not operational when it is not ready.

In response to an INQUIRY command the SCSI target device shall return the INQUIRY data with the peripheral qualifier set to the value specified in SPC-2. In response to a REQUEST SENSE command the SCSI target device shall return the REQUEST SENSE data with a sense key of NO SENSE.

The SCSI target device's response to any other command is vendor specific.

5.9.4 Sense data

5.9.4.1 Sense data introduction

Sense data shall be made available by the logical unit in the event a command completes with a CHECK CONDITION status or other conditions. The format, content and conditions under which sense data shall be prepared by the logical unit are specified in this standard, SPC-2, the applicable command set standard and applicable SCSI transport protocol standard.

Sense data shall be preserved by the logical unit for the SCSI initiator port until it is transferred by one of the methods listed below or until another task from that SCSI initiator port is entered into the task set.

The sense data may be transferred to the SCSI initiator device through any of the following methods:

a) The REQUEST SENSE command (see SPC-2);
b) An asynchronous event report (see 5.9.4.2); or
c) Autosense delivery (see 5.9.4.3).

5.9.4.2 Asynchronous event reporting

Asynchronous event reporting is used by a logical unit to signal another SCSI device that an asynchronous event has occurred. The mechanism automatically returns sense data associated with the event. A SCSI transport protocol standard may describe a mechanism for asynchronous event reporting. In this subclause, references to asynchronous event reporting assume that the SCSI device to be notified has enabled asynchronous event reports from the SCSI target device. Support for asynchronous event reporting is a logical unit option.

NOTE 9 - A SCSI device that is capable of producing asynchronous event reports at initialization time should provide means to disable generation of these reports. This may be done with a switch or jumper wire. SCSI devices that implement saved parameters may alternatively save the asynchronous event reporting permissions either on a per SCSI device basis or as a system wide option.

Parameters managing the use of asynchronous event reporting are contained in the Control mode page (see SPC-2).

Asynchronous event reporting is used to signal a SCSI device that one of the following events has occurred:

a) An exception condition was encountered after command completion;
b) A newly initialized device is available;
c) Some other type of unit attention condition has occurred; or
d) An asynchronous event has occurred.

An example of a) occurs in a SCSI target device that implements a write cache. If the SCSI target device is unable to write cached data to the medium, it may use an asynchronous event report to inform the SCSI initiator device of the failure.

An example of b) is a logical unit that generates an asynchronous event report, following a power-on cycle, to notify other SCSI devices that it is ready to accept I/O commands.

An example of c) occurs in a SCSI target device that supports removable media. Asynchronous event reporting may be used to inform a SCSI initiator device of a not-ready-to-ready transition (medium changed) or of an operator initiated event (e.g., activating a write protect switch or activating a start or stop switch).

An example of d) is a sequential-access device performing a REWIND command with the IMMEDIATE bit set to one (see SSC). An asynchronous event report may be used to inform a SCSI initiator device that the beginning of medium has been reached. Completion of a CD-ROM AUDIO PLAY command (see MMC-2) started in the immediate mode is another example of this case.

Sense data accompanying the asynchronous event report identifies the condition (see 5.9.4.1).

An exception condition encountered after command completion shall be reported to a specific SCSI initiator port once per occurrence of the event causing it. The logical unit may choose to use an asynchronous event report or to return CHECK CONDITION status on a subsequent command, but not both. Notification of an exception condition encountered after command completion shall be reported only to the SCSI initiator port or SCSI initiator ports that sent the affected task or tasks.

Asynchronous event reports may be used to notify SCSI devices that a system resource has become available. If a logical unit uses this method of reporting, the sense key in the asynchronous event report sense data shall be set to UNIT ATTENTION.

5.9.4.3 Autosense

Autosense is the automatic return of sense data to the application client coincident with the completion of a SCSI command under the conditions described in this subclause. All SCSI transport protocols shall support autosense.

If supported by the SCSI transport protocol and logical unit and requested by the **Execute Command** remote procedure call (see 5.1), the device server shall only return sense data in this manner coincident with the completion of a command with a status of CHECK CONDITION. After autosense data is sent the following shall be cleared:

a) The CA condition (see 5.9.1.6), if any; and
b) The sense data, except sense data associated with an unit attention condition when the UA_INTLCK_CTRL field in the Control mode page (see SPC-3) contains 10b or 11b.

Autosense shall not affect ACA (see 5.9.1) or the sense data associated with an unit attention condition when the UA_INTLCK_CTRL field contains 10b or 11b.

SCSI transport protocol standards that support autosense shall require an autosense implementation to:

a) Notify the logical unit when autosense data has been requested for a command; and
b) Inform the application client when autosense data has been returned upon command completion (see 5.1).

It is not an error for the application client to request the automatic return of sense data when autosense is not supported by the SCSI transport protocol or logical unit implementation. If the application client requested the return of sense data through the autosense facility and the SCSI transport protocol layer does not support this feature, then the confirmation returned by the SCSI initiator port should indicate that no sense data was returned. If the SCSI transport protocol layer supports autosense but the logical unit does not, then the SCSI target device should indicate that no sense data was returned. In either case, sense information shall be preserved and the application client may issue a command to retrieve it.

5.9.5 Unit Attention condition

Each logical unit shall generate an unit attention condition whenever the logical unit has been reset as described in 5.9.7 or by a power-on reset. In addition, a logical unit shall generate an unit attention condition for each SCSI initiator port whenever one of the following events occurs:

a) A removable medium may have been changed;
b) The mode parameters in effect for this SCSI initiator port have been changed by another SCSI initiator port (see SPC-2);
c) The version or level of microcode has been changed (see SPC-2);
d) Tasks for this SCSI initiator port were cleared by another SCSI initiator port;
e) INQUIRY data has been changed (see SPC-2);
f) The logical unit inventory has been changed (see SPC-2);
g) The mode parameters in effect for the SCSI initiator port have been restored from non-volatile memory (see SPC-2);
h) A change in the condition of a synchronized spindle; or
i) 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 power on condition followed by a microcode change condition).

An unit attention condition shall persist on the logical unit for each SCSI initiator port until that SCSI initiator port clears the condition as described in the remainder of this subclause.

If an INQUIRY command enters the enabled task state, the logical unit shall perform the INQUIRY command and shall neither report nor clear any unit attention condition.

If a REPORT LUNS command enters the enabled task state, the logical unit shall perform the REPORT LUNS command and shall not report any unit attention condition. The logical unit shall clear any unit attention condition established in response to a change in the logical unit inventory for all logical units for the SCSI initiator port that sent the REPORT LUNS command. The logical unit shall not clear any other unit attention condition.

If a REQUEST SENSE command enters the enabled task state while an unit attention condition exists for the SCSI initiator port that sent the REQUEST SENSE command, then the logical unit shall either:

a) Report any pending sense data and preserve all unit attention conditions on the logical unit; or,
b) Report an unit attention condition for the SCSI initiator port that sent the REQUEST SENSE command. The logical unit may discard any pending sense data and shall clear the reported unit attention condition for that SCSI initiator port.

If the logical unit has already generated the CA or ACA condition for an unit attention condition, the logical unit shall report the unit attention condition (i.e., option b) above).

If a command other than INQUIRY, REPORT LUNS, or REQUEST SENSE enters the enabled task state while an unit attention condition exists for the SCSI initiator port that sent the command, the logical unit shall terminate the command with a CHECK CONDITION status. The logical unit shall provide sense data that reports an unit attention condition for the SCSI initiator port that sent the command.

If a logical unit reports an unit attention condition with autosense (see 5.9.4.3) or with an asynchronous event report (see 5.9.4.2) and the UA_INTLCK_CTRL field in the Control mode page contains 00b (see SPC-3), then the logical unit shall clear the reported unit attention condition for that SCSI initiator port on the logical unit. If the UA_INTLCK_CTRL field in the Control mode page contains 10b or 11b, the logical unit shall not clear unit attention conditions reported with autosense or an asynchronous event report.

5.9.6 Hard reset

A hard reset is a SCSI target port action in response to a reset event within the service delivery subsystem. A wakeup event (see 3.1.132) is a reset event. The definition of additional reset events is SCSI transport protocol specific. Each SCSI transport protocol standard that defines reset events shall specify the SCSI target port's action in response to reset events.

The SCSI target port's response to a hard reset shall include initiating the equivalent of a logical unit reset for all logical units as described in 5.9.7.

While 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 in response to a reset event within the service delivery subsystem shall be unaffected by access controls.

5.9.7 Logical unit reset

A logical unit reset is:

a) The action in response to a LOGICAL UNIT RESET task management request (see 6.6) or some other logical unit reset event; or
b) One of the actions in response to a TARGET RESET task management function (see 6.8) or a hard reset (see 5.9.6).

The definition of logical unit reset events is dependent on the SCSI transport protocol.

To process a logical unit reset the logical unit shall:

a) Abort all tasks as described in 5.7;
b) Clear a CA (see 5.9.1.6) or ACA (see 5.9.1.7) condition, if one is present;
c) Release all reservations established using the reserve/release management method (persistent reservations shall not be affected);
d) Return the logical unit's operating mode to the appropriate initial conditions, similar to those conditions that would be found following device power-on. The MODE SELECT parameters (see SPC-2) shall be restored to their last saved values if saved values have been established. MODE SELECT parameters for which no saved values have been established shall be returned to their default values;
e) Set an unit attention condition (see 5.9.5); and
f) Initiate a logical unit reset for all dependent logical units (see 4.13).

In addition to the above, the logical unit shall perform any additional functions required by the applicable standards.

6 Task management functions

6.1 Introduction

Task management functions control the processing of one or more tasks. An application client requests a task management function by means of a procedure call having the following format:

Service Response = Function name (IN (nexus))

Service Response:

One of the following SCSI transport protocol specific 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: An optional task manager response indicating that the operation is supported and completed successfully. This task manager response shall only used by operations that require notification of success.

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

SERVICE DELIVERY OR TARGET FAILURE: The request was terminated due to a service delivery failure (see 3.1.108) 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 actual events comprising each of the above service responses.

The task management functions are summarized in table 31.

Table 31 — Task Management Functions

Task Management Function	Nexus	Reference
ABORT TASK	I_T_L_Q	6.2
ABORT TASK SET	I_T_L	6.3
CLEAR ACA	I_T_L	6.4
CLEAR TASK SET	I_T_L	6.5
LOGICAL UNIT RESET	I_T_L	6.6
QUERY TASK	I_T_L_Q	6.7
TARGET RESET	I_T	6.8
WAKEUP	I_T	6.9

Argument descriptions:

Nexus: An I_T Nexus, I_T_L Nexus, or I_T_L_Q Nexus (see 4.11).

I_T Nexus: A SCSI initiator port and SCSI target port nexus (see 4.11).

I_T_L Nexus: A SCSI initiator port, SCSI target port, and logical unit nexus (see 4.11).

I_T_L_Q Nexus: A SCSI initiator port, SCSI target port, logical unit, and tag nexus (see 4.11).

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, or QUERY TASK shall not be affected by the presence of access restrictions;
b) A task management request of CLEAR TASK SET or LOGICAL UNIT RESET received from a SCSI initiator port that is denied access to the logical unit (either because it has no access rights or because it is in the pending-enrolled state) shall cause no change to the logical unit;
c) A TARGET RESET task management request shall initiate a logical unit reset as described in 5.9.7 for all logical units to which the SCSI initiator port has access, and shall cause no change to any logical units to which the SCSI initiator port is denied access; and
d) The task management function Service Response shall not be affected by the presence of access restrictions.

6.2 ABORT TASK

Function call:

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

Description:

This function shall be supported by a logical unit if it supports tagged tasks and may be supported by a logical unit if it does not support tagged tasks.

The task manager shall abort the specified task, if it exists, as described in 5.7. Previously established conditions, including MODE SELECT parameters, reservations, CA, and ACA shall not be changed by the ABORT TASK function.

If the logical unit supports this 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 responses from the task are sent to the SCSI initiator port.

All SCSI transport protocol standards shall provide the functionality needed for a task manager to implement the ABORT TASK task management function.

6.3 ABORT TASK SET

Function Call:

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

Description:

This function shall be supported by all logical units.

The task manager shall abort all tasks in the task set that were created by the SCSI initiator port routed through the SCSI target port as described in 5.7.

The task manager shall perform an action equivalent to receiving a series of ABORT TASK requests. All tasks from that SCSI initiator port in the task set shall be aborted. Tasks from other SCSI initiator ports or in other task sets shall not be aborted. A CA shall be cleared by the ABORT TASK SET function from the faulted initiator port (see 5.9.1.6). 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 provide the functionality needed for a task manager to implement the ABORT TASK SET task management function.

6.4 CLEAR ACA

Function Call

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

Description:

This function shall be supported by a logical unit if it supports ACA (see 5.2.3).

The application client issues CLEAR ACA to clear an ACA condition from the task set serviced by the logical unit as specified in 5.9.1.7. For tasks with the ACA attribute (see 7.5.4) receipt of an CLEAR ACA function shall have the same effect as receipt of an ABORT TASK function (see 6.2). 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 7.

While a CA is in effect (see 5.9.1), a logical unit that supports the CLEAR ACA task management function shall ignore all CLEAR ACA requests and shall return a service response of FUNCTION COMPLETE.

All SCSI transport protocol standards shall provide the functionality needed for a task manager to implement the CLEAR ACA task management function.

6.5 CLEAR TASK SET

Function Call:

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

Description:

This function shall be supported by all logical units, except in the following cases, when support for this function is optional:

a) The logical unit does not support tagged tasks (see 4.10); or
b) The logical unit supports the basic task management model (see 7.2).

All tasks in the appropriate task set as defined by the TST field in the Control mode page (see SPC-2) shall be aborted as described in 5.7. The medium may have been altered by partially processed commands.

The CA condition (see 5.9.1.6), and all pending status and sense data for the task set defined by the TST field in the Control mode page 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 provide the functionality needed for a task manager to implement the CLEAR TASK SET task management function.

6.6 LOGICAL UNIT RESET

Function Call:

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

Description:

This function shall be supported by all logical units.

Before returning a FUNCTION COMPLETE response, the logical unit shall perform the logical unit reset functions specified in 5.9.7.

NOTE 10 - 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 provide the functionality needed for a task manager to implement the LOGICAL UNIT RESET task management function.

6.7 QUERY TASK

Function call:

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

Description:

SCSI transport protocols may or may not support QUERY TASK and may or may not require logical units accessible through target ports using such transport protocols to support QUERY TASK. The task manager shall return a response of FUNCTION SUCCEEDED if the specified task exists, or FUNCTION COMPLETE if the specified task does not exist.

6.8 TARGET RESET

Function Call:

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

Description:

Before returning a FUNCTION COMPLETE response, the SCSI target port shall cause logical unit reset functions to be performed as specified in 5.9.7 for every logical unit.

An application client should issue LOGICAL UNIT RESETs only to the logical units it is using rather than issuing a TARGET RESET. This avoids resetting logical units that other SCSI initiator ports, possibly in other SCSI initiator devices, may be using.

NOTE 11 - Previous versions of this standard required TARGET RESET support in all SCSI target devices. SCSI transport protocols may or may not require that TARGET RESET be supported. SCSI transport protocols may require additional actions beyond those specified here.

6.9 WAKEUP

Function Call:

Service response = WAKEUP (IN (I_T Nexus))

Description:

SCSI transport protocols may or may not define the WAKEUP function. This function may be supported by SCSI transport protocols whose interconnects support a shared wakeup signal or individual wakeup signals for each SCSI target port. This function may be supported by SCSI devices on SCSI transport protocols that support the function.

This function causes a wakeup event (see SPC-3) to be sent to either:

a) The specified SCSI target port, on SCSI transport protocols supporting individual wakeup signals; or
b) All SCSI target ports connected to the interconnect, on SCSI transport protocols supporting a shared wakeup signal.

The wakeup function is a reset event and shall cause a hard reset in the recipient SCSI target port(s).

6.10 Task management SCSI transport protocol services

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 remote procedure call. The following arguments are passed:

Nexus: An I_T Nexus, I_T_L Nexus, or I_T_L_Q Nexus (see 4.11).

Function Identifier: Parameter 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** SCSI transport protocol service and the **Received Task Management Function Executed** confirmation described below. Support for the **Task Management Request Received** indication and **Task Management Function Executed** SCSI transport protocol service response by the SCSI transport protocol standard is optional. All SCSI devices shall implement these SCSI transport protocol services as defined in the applicable SCSI transport protocol standard.

Request sent by an application client to a task manager:

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

Argument descriptions:

Nexus: An I_T Nexus, I_T_L Nexus, or I_T_L_Q Nexus (see 4.11).

Function Identifier: Parameter encoding the task management function to be performed.

Indication received by the task manager:

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

Argument descriptions:

Nexus: An I_T Nexus, I_T_L Nexus, or I_T_L_Q Nexus (see 4.11).

Function Identifier: Parameter encoding the task management function to be performed.

Response from task manager to application client:

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

Argument descriptions:

Nexus: An I_T Nexus, I_T_L Nexus, or I_T_L_Q Nexus (see 4.11).

Service Response: An encoded value representing one of the following:

FUNCTION COMPLETE: The requested function has been completed.

FUNCTION REJECTED: The task manager does not implement the requested function.

SERVICE DELIVERY OR TARGET FAILURE: The request was terminated due to a service delivery failure (see 3.1.111) or SCSI target device malfunction. The task manager may or may not have successfully performed the specified function.

Confirmation received by application client:

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

Argument descriptions:

Nexus: An I_T Nexus, I_T_L Nexus, or I_T_L_Q Nexus (see 4.11).

Service Response: An encoded value representing one of the following:

FUNCTION COMPLETE: The requested function has been completed.

FUNCTION REJECTED: The task manager does not implement the requested function.

SERVICE DELIVERY OR TARGET FAILURE: The request was terminated due to a service delivery failure (see 3.1.111) or SCSI target device malfunction. The task manager may or may not have successfully performed the specified function.

Since the nexus used by all task management functions except ABORT TASK does not contain a task tag to uniquely identify the transaction, there may be no way for an application client to associate a confirmation with a request. A SCSI transport protocol that does not provide such an association should not allow a SCSI initiator port to have more than one pending task management request per I_T_L nexus.

6.11 Task management function example

Figure 31 shows the sequence of events associated with a task management function.

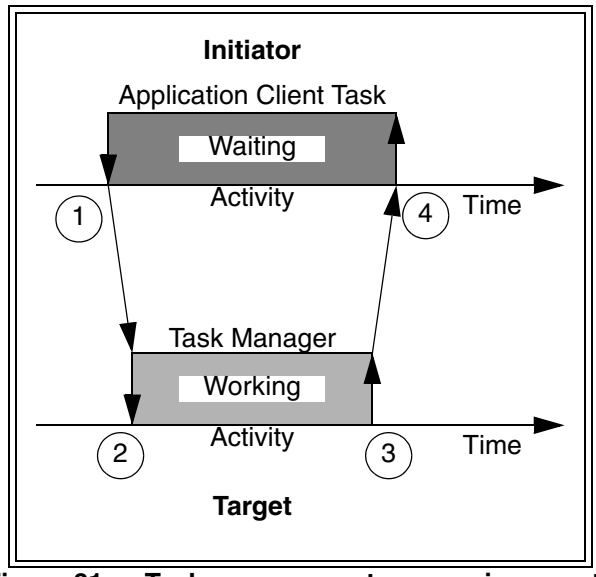


Figure 31 — Task management processing events

The numbers in figure 31 identify the events described below.

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 operation and responds by invoking the **Task Management Function Executed** SCSI transport protocol service to notify the application client. The service response parameter is set to a value of FUNCTION COMPLETE.

4. A **Received Task Management Function Executed** confirmation is received by the application client task.

7 Task Set Management

7.1 Introduction to task set management

Clause 7 describes some of the controls application clients have over task set management behaviors (see 7.2). Clause 7 also specifies task set management requirements in terms of:

a) Task states (see 7.4);
b) Task attributes (see 7.5);
c) The events that cause transitions between task states (see 7.3 and 7.4); and
d) A map of task state transitions (see 7.6).

Clause 7 concludes with several task set management examples (see 7.7).

Task behavior, as specified in clause 7, 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 5.9.2) 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 a status other than GOOD.

7.2 Controlling task set management

The Control mode page (see SPC-2) contains fields that specify particular task set management behaviors. The standard INQUIRY data CmdQue bit (see SPC-2) indicates support for tagged tasks (command queuing). One specific combination of task set management behaviors is identified as the basic task management model. Support for the basic task management model is indicated by values returned in the CMDQUE and BQUE bits in the standard INQUIRY data (see SPC-2). The basic task management model requires the following task set management behaviors:

a) The only task attribute supported shall be SIMPLE;
b) The device server may reorder the actual processing sequence of tasks in any manner. Any data integrity exposures related to task sequence order shall be explicitly handled by the application client using the appropriate commands;
c) All the tasks shall be aborted when an CA or ACA condition is established;
d) It shall not be possible to disable tagged queuing; and
e) Support for the CLEAR TASK SET task management function is optional.

7.3 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-2) equals 000b, all tasks have ended that were accepted from all SCSI initiator ports earlier in time than the referenced task. If the TST field in the Control mode page equals 001b, all tasks have ended that were accepted from the referenced SCSI initiator port earlier in time than the referenced task.

All older head of queue and older ordered tasks ended: If the TST field in the Control mode page equals 000b, all head of queue and ordered tasks have ended that were accepted from all SCSI initiator ports earlier in time than the referenced task. If the TST field in the Control mode page equals 001b, all head of queue and ordered tasks have ended that were accepted from the referenced SCSI initiator port earlier in time than the referenced task.

CA or ACA establishment: A CA or ACA condition has been established (see 5.9.1).

task abort: A task has been aborted as described in 5.7.

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.

CA cleared: An CA condition has been cleared (see 5.9.1.6).

ACA cleared: An ACA condition has been cleared (see 5.9.1.7).

7.4 Task states

7.4.1 Overview

7.4.1.1 Task state nomenclature

The model employs four tasks states, summarized in table 32.

Table 32 — Task State Nomenclature

Task State Name	Reference	Tasks in This State May Be Called
Enabled task state	7.4.2	Enabled tasks
Blocked task state	7.4.3	Blocked tasks
Dormant task state	7.4.4	Dormant tasks
Ended task state	7.4.5	Ended tasks

7.4.1.2 Suspended information

Any information the logical unit has or accepts for a task the blocked task state (see 7.4.3) or dormant task state (see 7.4.4) is required to be held in a condition where it is not available to the task. Such information is called, suspended information.

7.4.2 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-2). 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. Although, before entering this state for the first time, the task may perform other activities visible to lower layers – such as pre-fetching data to be written to the media – 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.

7.4.3 Blocked task state

A task in the blocked task state is prevented from completing due to an CA or 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-2) equals 000b the blocked task state is independent of the SCSI initiator port. If the TST field equals 001b the blocked task state applies only to the faulted initiator port.

7.4.4 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.

7.4.5 Ended task state

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

7.4.6 Task states and task lifetimes

Figure 32 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.

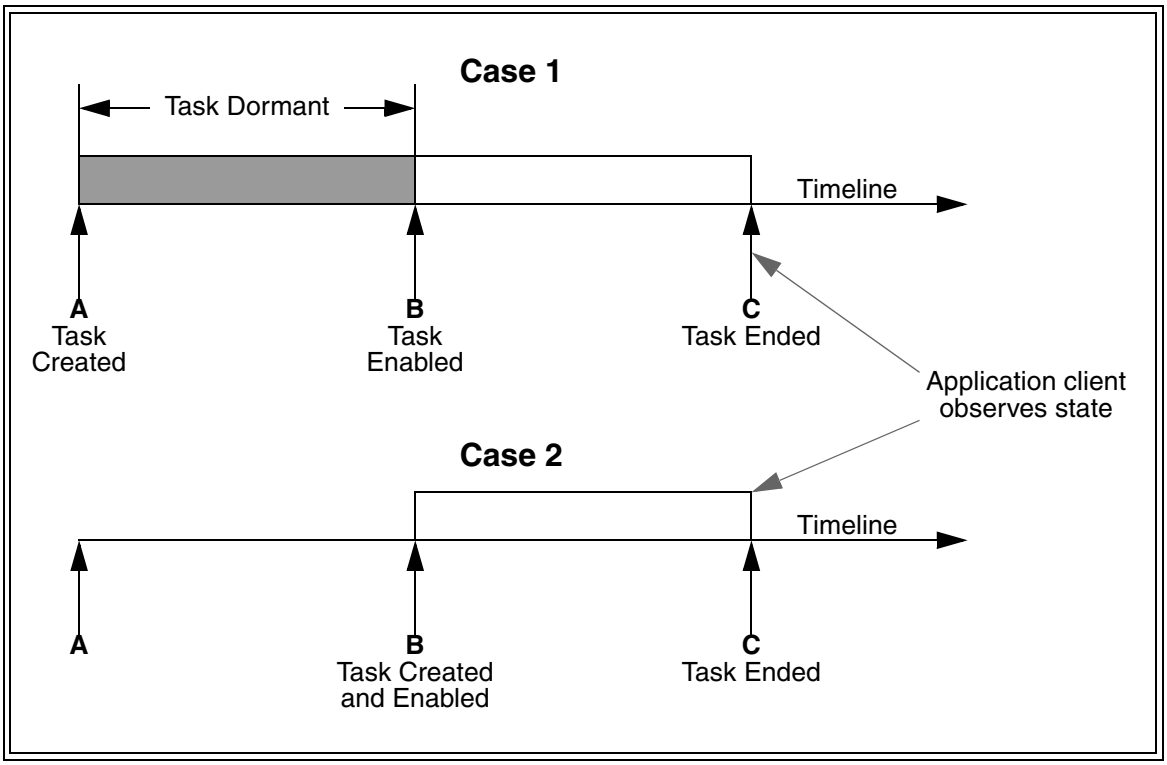


Figure 32 — Example of Dormant state task behavior

7.5 Task attributes

7.5.1 Simple task

A task having the SIMPLE attribute shall be accepted into the task set in the dormant task state. The task shall not enter the enabled task state until all older head of queue and older ordered tasks in the task set have ended (see 7.3).

7.5.2 Ordered task

A task having the ORDERED attribute shall be accepted into the task set in the dormant task state. The task shall not enter the enabled task state until all older tasks in the task set have ended (see 7.3).

7.5.3 Head of queue task

A task having the HEAD OF QUEUE attribute shall be accepted into the task set in the enabled task state.

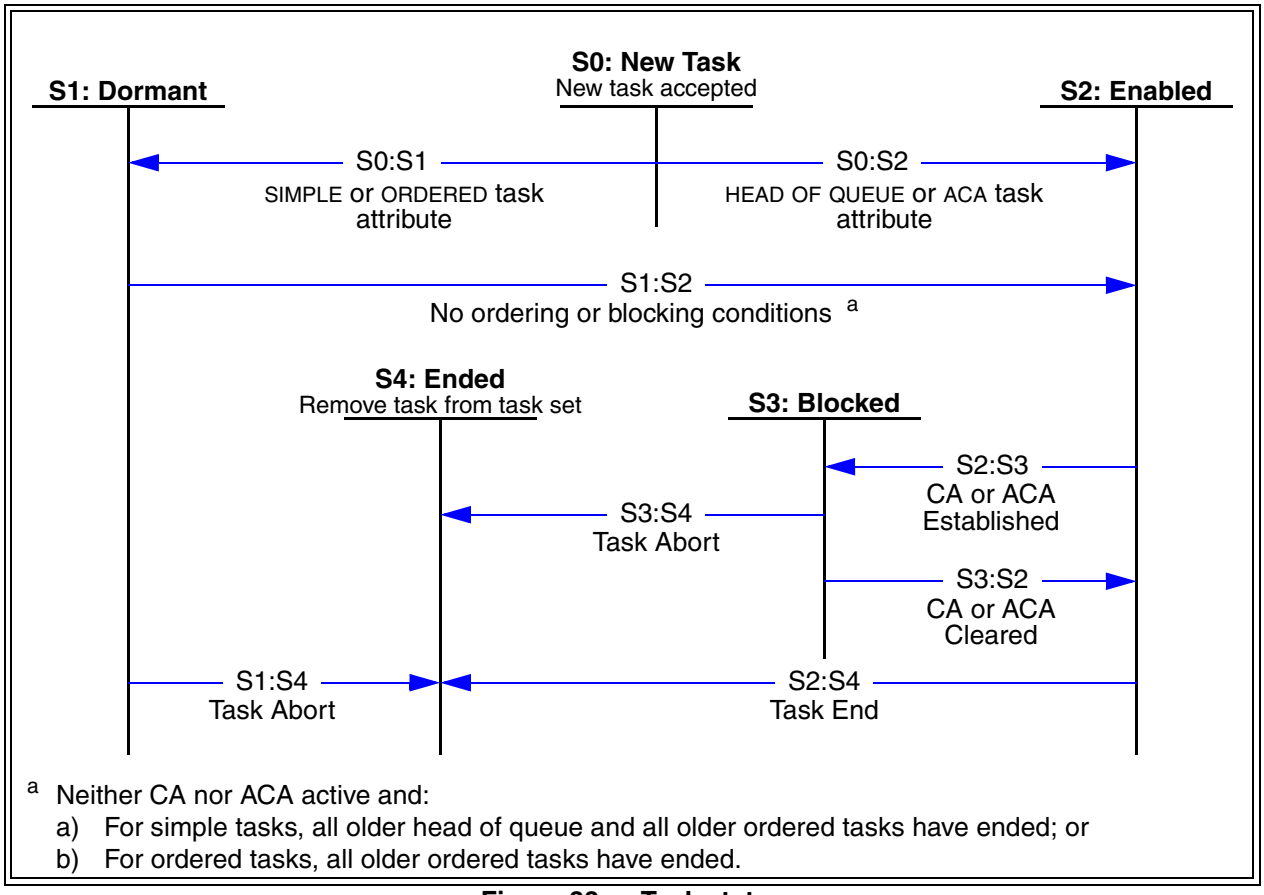
7.5.4 ACA task

A task having the ACA attribute shall be accepted into the task set in the enabled task state. There shall be no more than one ACA task per task set (see 5.9.1.2).

7.6 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 33 shows the behavior of a single task in response to an external event.



a Neither CA nor ACA active and:
a) For simple tasks, all older head of queue and all older ordered tasks have ended; or
b) For ordered tasks, all older ordered tasks have ended.

Figure 33 — 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 older head of queue and older ordered tasks (see 7.3) have ended; or
b) A dormant task having the ORDERED task attribute shall enter the enabled task state when all older tasks (see 7.3) have ended.

If the TST field in the Control mode page (see SPC-2) contains 000b, then the transition from dormant task to enabled task shall not occur while a CA or ACA is in effect for any SCSI initiator port (see 5.9.1.4 and 5.9.1.5). If the TST field in the Control mode page contains 001b, then dormant tasks from the faulted initiator port shall not transition to the enabled task state while a CA or ACA is in effect for that SCSI initiator port (see 5.9.1.4).

Transition S2:S3: The establishment of a CA or ACA condition (see 7.3) shall cause zero or more enabled tasks to enter the blocked task state as described in 5.9.1.2.

Transition S3:S2: When a CA or ACA condition is cleared (see 7.3), tasks that entered the blocked state the CA or ACA condition was established (see 5.9.1.2) shall re-enter the enabled task state.

Transition S2:S4: A task that has completed (see 7.3) or aborted (see 7.3 and 5.7) shall enter the ended task state. This is the only state transition that applies to an ACA task.

Transitions S1:S4, S3:S4: A task abort event (see 7.3 and 5.7) shall cause the task to unconditionally enter the ended task state.

7.7 Task set management examples

7.7.1 Introduction

Several task set management scenarios are shown in 7.7.2, 7.7.3, and 7.7.4. 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 SCSI initiator port originating a task). The examples are also valid for a single SCSI initiator port, when the TST field contains 001b (i.e., task set management proceeds independently for each SCSI initiator port and the events and transitions in one SCSI initiator port's task set do not affect the task set management for another SCSI initiator port's task set). 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-2).

The figure accompanying each example shows successive snapshots of a task set after various events, such as task creation or completion. In all cases, the constraints on task completion order established using the QUEUE ALGORITHM MODIFIER field and DQUE bit in the Control mode page (see SPC-2) 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 attributes are shown in table 33.

Table 33 — Task attribute and state indications in examples

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

The conditions preventing a dormant task from entering enabled task state (except for CA and 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 no CA or ACA is in effect, a task enters the enabled task state after all intervening barriers have been removed.

7.7.2 Head of queue tasks

Figure 34 shows task set conditions when several head of queue tasks are processed.

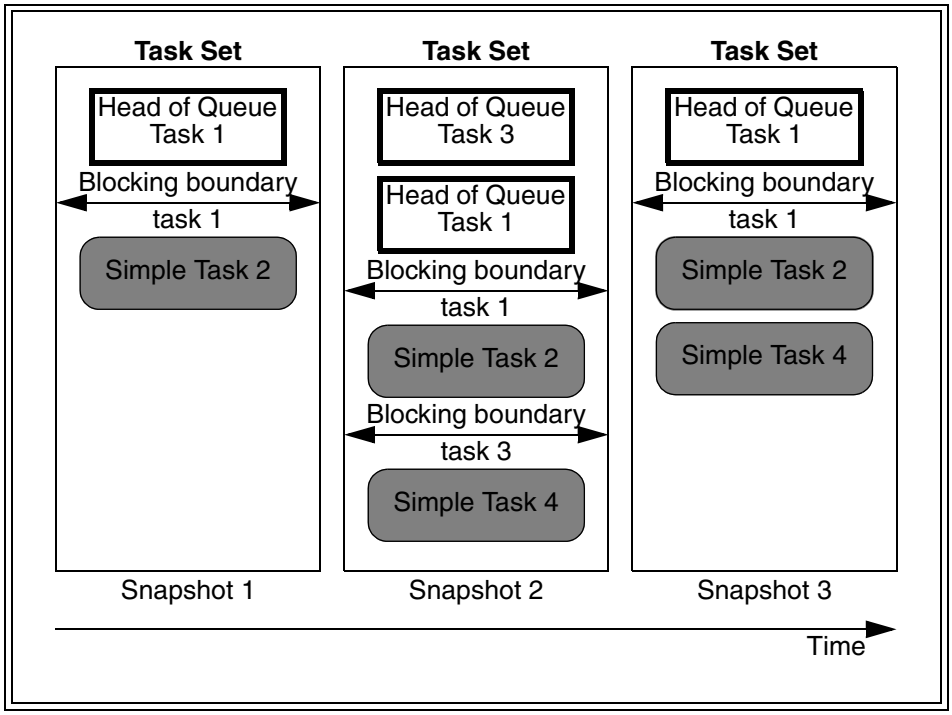


Figure 34 — Head of queue tasks and blocking boundaries (example 1)

In snapshot 1 the task set initially 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 older 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, tasks 2 and 4 remain in the dormant task state.

Figure 35 is the same as the previous example, except that task 1 completes instead of task 3.

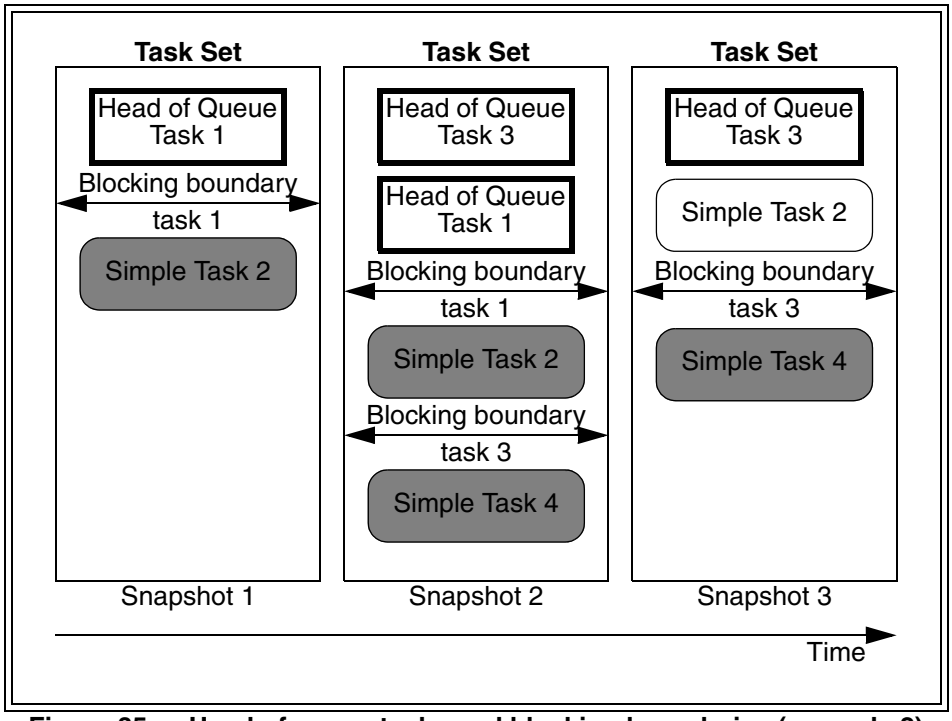


Figure 35 — Head of queue tasks and blocking boundaries (example 2)

The completion of task 1 allows task 2 to enter the enabled task state. Simple task 4 is placed in the dormant task state until task 3 completes.

7.7.3 Ordered tasks

An example of ordered and simple task interaction is shown in figure 36.

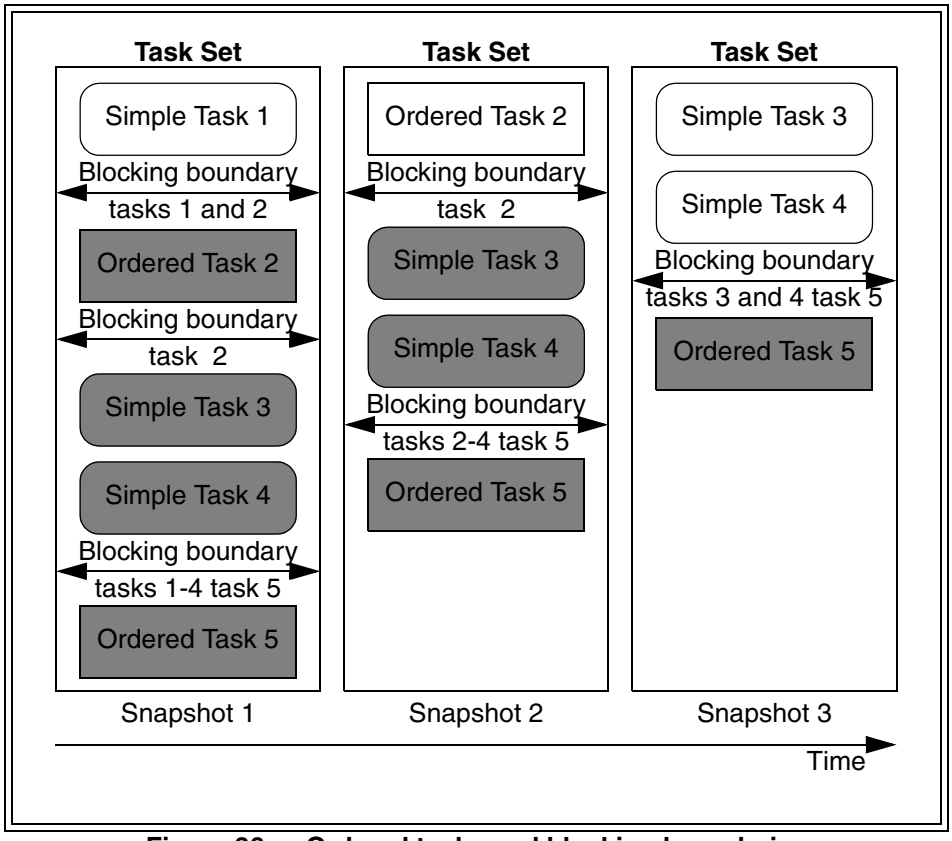


Figure 36 — Ordered tasks and blocking boundaries

The state of dormant tasks 2 through 5 is determined by the requirements shown in table 34.

Table 34 — 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 older tasks have ended.
5	
3	A simple task is not allowed to enter the enabled task state until all older head of queue and older ordered tasks have ended.
4	

The table 34 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 tasks 3, 4 and 5 are still in effect, these tasks must remain in the dormant task state. As shown in snapshot 3, the completion of task 2 triggers two state changes: the transitions of task 3 and task 4 to the enabled task state. Task 5 must remain in the dormant task state until these tasks end.

7.7.4 ACA task

Figure 37 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-2). Consequently, clearing an ACA condition does not cause tasks to be aborted.

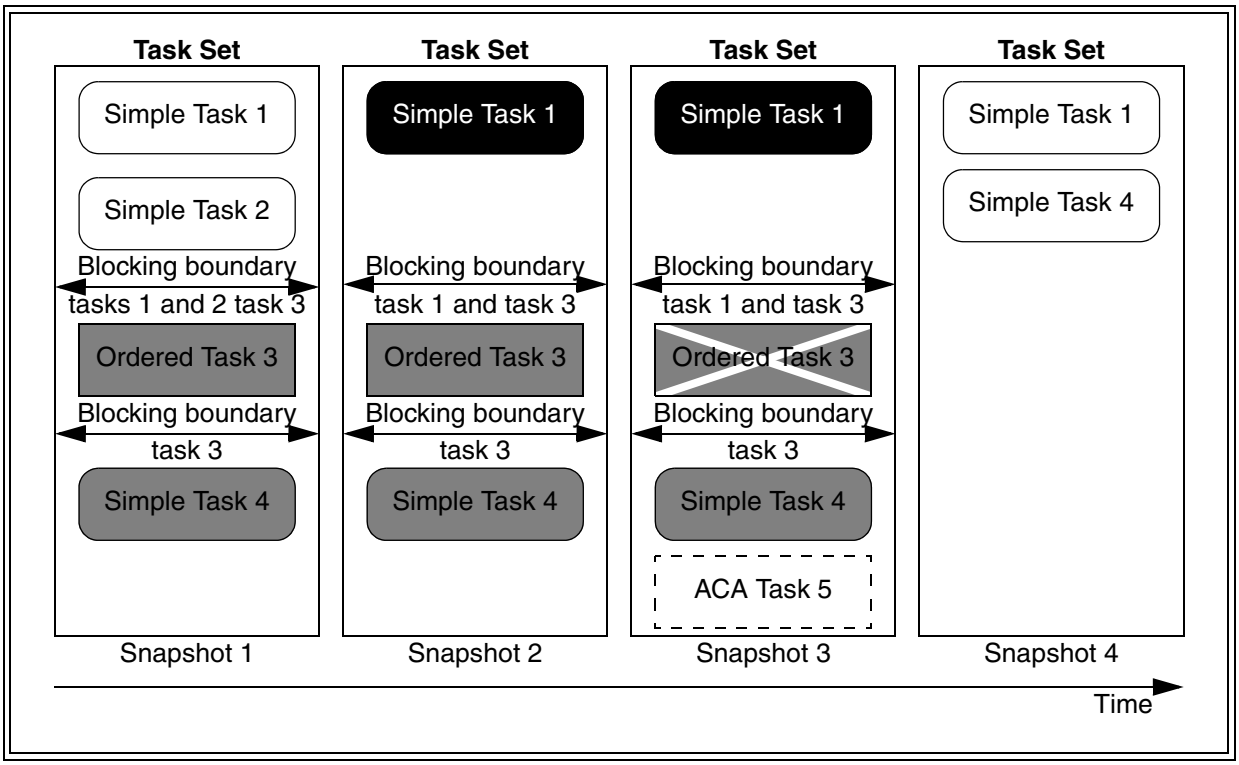


Figure 37 — ACA task example

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, (snapshot 4) simple task 1 is allowed to reenter the enabled task state. Since there are no head of queue or ordered tasks older than task 4, it too is allowed enter the enabled task state.

Annex A
(informative)

Identifiers and names for objects

A.1 Identifiers and names overview

This annex summarizes SCSI identifiers and names.

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

a) SCSI initiator port (see 3.1.90);
b) SCSI target port (see 3.1.100);
c) Logical unit (see 3.1.56);
d) SCSI initiator device (see 3.1.89); and
e) SCSI target device (see 3.1.99).

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-2 requires SCSI transport protocols and logical units to support identifiers and names for an object.

Table A.1 — 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	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-2).

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

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

Object	Identifier size					
	SPI-4	FCP-2	SRP	iSCSI	SBP-3	SAS SSP
Initiator port	4 bits [a]	3 bytes	16 bytes	255 bytes	2 bytes	8 bytes
Target port	4 bits [a]	3 bytes	16 bytes	255 bytes	11 bytes	8 bytes
Logical unit	6 bits (data group transfers) 8 bytes (information unit transfers)	8 bytes	8 bytes	8 bytes	2 bytes	8 bytes

[a] SPI-4 uses a bit significant representation of the SCSI port identifier, therefore, the maximum number of SCSI ports is 16, a value that can be represented in 4 bits.

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

Object	Identifier format					
	SPI-4	FCP-2	SRP	iSCSI	SBP-3	SAS SSP
Initiator port	bit significant (a maximum of 16 ports; one for each bit)	binary value	EUI-64 + 8 byte extension [a]	iSCSI name + “i” + Initiator Session Identifier (ISID) [b] [c]	binary value	NAA IEEE Registered format per SPC-3
Target port	bit significant (a maximum of 16 ports; one for each bit)	binary value	EUI-64 + 8 byte extension [a]	iSCSI name + “t” + Target Portal Group Tag [b] [d]	EUI-64 + Discovery ID [e]	NAA IEEE Registered format per SPC-3
Logical unit	binary value (6 bit) or As specified in this standard (8 byte)	As specified in this standard	As specified in this standard	As specified in this standard	As specified in this standard	As specified in this standard

[a] Required to be worldwide unique and recommend to be EUI-64 + 8 byte extension.
[b] The iSCSI name should be worldwide unique, 223 bytes maximum in UTF-8 format with null termination.
[c] The Initiator Session Identifier (ISID) is a non-zero six byte integer.
[d] The Target Portal Group Tag is a non-zero two byte integer.
[e] See IEEE Std P1212 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. See table A.5 for a list of the format of the names for each SCSI transport protocol.

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

Object	Name size					
	SPI-4	FCP-2	SRP	iSCSI	SBP-3	SAS SSP
Initiator device	not specified	not specified	not specified	223 bytes	not specified	not specified
Target device	not specified	not specified	not specified	223 bytes	not specified	not specified
Initiator port	not specified	8 bytes	16 bytes	255 bytes	8 bytes	not specified
Target port	not specified	8 bytes	16 bytes	255 bytes	11 bytes	not specified
Logical unit	Reported in the Device Identification VPD page (see SPC-2).					

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

Object	Name format					
	SPI-4	FCP-2	SRP	iSCSI	SBP-3	SAS SSP
Initiator device	not specified	not specified	not specified	iSCSI name [a]	not specified	not specified
Target device	not specified	not specified	not specified	iSCSI name [a]	not specified	not specified
Initiator port	not specified	Fibre Channel name_ identifier	EUI-64 + 8 byte extension [b]	iSCSI name + “i” + Initiator Session Identifier (ISID) [a] [c]	EUI-64	not specified
Target port	not specified	Fibre Channel name_ identifier	EUI-64 + 8 byte extension [b]	iSCSI name + “t” + Target Portal Group Tag [a] [d]	EUI-64 + Discovery ID [e]	not specified
Logical unit	Device Identification VPD page name (see SPC-2)					

[a] The iSCSI name should be worldwide unique, 223 bytes maximum in UTF-8 format with null termination.
[b] Required to be worldwide unique and recommend to be EUI-64 + 8 byte extension.
[c] The Initiator Session Identifier (ISID) is a non-zero six byte integer.
[d] The Target Portal Group Tag is a non-zero two byte integer.
[e] See IEEE Std P1212 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 IEEE Std P1212: Standard for a Control and Status Register (CSR) Architecture for Microcomputer Buses. See http://www.ieee.org/.

A.3.4 iSCSI: As of this writing, the most recently published iSCSI internet draft is: http://www.ietf.org/internet-drafts/draft-ietf-ips-iscsi-16.txt. Newer drafts may be identified at http://http://www.ietf.org/html.charters/ips-charter.html. The iSCSI internet draft is a standards track RFC specification.

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 SPI-4: SCSI Parallel Interface -4 (see 1.3).

A.3.10 SRP: SCSI RDMA Protocol (see 1.3).

A.3.11 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/.

Annex B
(informative)

Terminology mapping

The introduction of a model for SCSI devices with multiple ports resulted in changes in terminology between SAM and SAM-2 (see table B.1).

Table B.1 — SAM-2 to SAM terminology mapping

SAM-2 equivalent term	SAM term
initiator port identifier	initiator identifier
SCSI initiator port	initiator
SCSI port	port
SCSI port identifier	device identifier
SCSI port identifier	SCSI identifier
SCSI target port	target
target port identifier	target identifier