Working Draft American National Standard

Project T10/2104-D

Revision 13
January 23, 2013

Information technology - SCSI Architecture Model - 5 (SAM-5)

This is an internal working document of T10, a Technical Committee of Accredited Standards Committee INCITS (International Committee for Information Technology Standards). As such this is not a completed standard and has not been approved. The contents may be modified by the T10 Technical Committee. The contents are actively being modified by T10. This document is made available for review and comment only.

T10 Technical Editor: George Penokie
LSI Corporation
3033 41st Street NW, Suite 100
Rochester, MN 55901
USA

Telephone: (507) 328-9017
Email: george.penokie@lsi.com

Reference number
ISO/IEC 14776-415:201x
ANSI INCITS *-201x**

Printed 10:56 AM Wednesday 16 January 2013

Points of Contact

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

T10 Chair
John B. Lohmeyer
LSI Corporation
4420 ArrowsWest Drive
Colorado Springs, CO 80907-3444
USA

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

T10 Vice-Chair
Mark S. Evans
Western Digital Corporation
5863 Rue Ferrari
San Jose, CA 95138
USA

Telephone: (408) 363-5257
Email: mark.evans@wdc.com

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

INCITS Secretariat
Suite 610
1101 K Street, NW
Washington, DC 20005
USA

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

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

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

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

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

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

American National Standard
for Information Technology

SCSI Architecture Model - 5 (SAM-5)

Secretariat
Information Technology Industry Council

Approved mm.dd.yy

American National Standards Institute, Inc.

ABSTRACT

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

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 towards their resolution.

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

The American National Standards Institute does not develop standards and will in no circumstances give interpretation on any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an 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 patents, if any, may apply to this standard. As of the date of publication of this standard, 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 W. 42nd Street, New York, New York 10036

Printed in the United States of America

Revision Information

R.1 Revision SAM-5r00 (24 September 2008)

First release of SAM-5. Same content as SAM-4 revision 14.

R.2 Revision SAM-5r01 (28 January 2009)

The following T10 approved proposals were incorporated in SAM-5 in this revision:

a) 08-327r2 - Restrict SPC-4 and SAM-5 codes to FC-SB-4 use [Butt];
b) 08-450r1 - Add Restricted keyword to the Style Guide [Weber];
c) 09-003r0 - Correct redundant text in command state transition figure [Knight]; and
d) 09-017r1 - QUERY TASK TMF Progress Indicator [Knight].

R.3 Revision SAM-5r02 (22 July 2009)

The following T10 approved proposals were incorporated in SAM-5 in this revision:

a) 09-159r1 - SAM5 - CLEAR_ACA status when no ACA condition exists [Knight];
b) An error was corrected that was induced in SAM-4 that was pointed out in the following e-mail from Rob Elliott on 03/05/2009:

   “SAM-4 no longer has 06-411r2 (REPORTED LUNS DATA HAS CHANGED unit attention clearing) incorporated correctly. In 5.14, items c) and d) are supposed to be paragraphs inside b), as they only apply to the REPORT LUNS command.

   c) if the UA_INTLCK_CTRL field in the Control mode page is set to 00b (see SPC-4), the SCSI target device shall clear any pending unit attention condition with an additional sense code of REPORTED LUNS DATA HAS CHANGED established for the SCSI initiator port associated with that I_T nexus in each logical unit accessible by the I_T nexus on which the REPORT LUNS command was received. Other pending unit attention conditions shall not be cleared;

   d) if the UA_INTLCK_CTRL field in the Control mode page contains 10b or 11b, the SCSI target device shall not clear any unit attention condition(s);

   i) if a device server reports a unit attention condition with CHECK CONDITION status and the UA_INTLCK_CTRL field in the Control mode page contains 00b (see SPC-4), then the device server shall clear the reported unit attention condition for the SCSI initiator port associated with that I_T nexus on the logical unit. If the unit attention condition has an additional sense code of REPORTED LUNS DATA HAS CHANGED, the SCSI target device shall clear any pending unit attention conditions with an additional sense code of REPORTED LUNS DATA HAS CHANGED established for the I_T nexus on which the command was received in each logical unit accessible by that I_T nexus. If the UA_INTLCK_CTRL field contains 10b or 11b, the device server shall not clear unit attention conditions reported with CHECK CONDITION status.

   Everything was correct from sam4r08 through sam4r13c, where this section was in section 5.8.7. It was broken in sam4r14d when the section was renumbered to 5.14. It looks like every top level paragraph in that list got changed to an UnorderedList0, including the unnumbered ones. f) and i) are also wrong.”

c) An error in a cross reference between SAM-5 and SPC-4 was pointed out in the following e-mail from Fred Knight on 03/09/2009.

   “SAM5r1 4.5.4.2 - SCSI Device Name attribute - last paragraph -...may be reported in a target device name designation descriptor in the Device Identification VPD page (see SPC-4).

   SPC4r18 - target device name - doesn't exist/.”

   This was corrected by changing the reference in subclause 4.5.4.2 SCSI Device Name attribute and subclause 4.5.25 Well Known Logical Unit class to the following:

   “in the Device Identification VPD pages designation descriptor for SCSI target devices (see SPC-4).”

d) An error in the incorporation of 09-017r1 was pointed out by Mark Evens in the following e-mail:

“While going through SAM-5r01 I noticed that you put the new Additional Response Information argument definition in QUERY TASK SET. Isn’t this supposed to be in QUERY TASK?”

The answer is yes it should have been in QUERY TASK.

e) Moved section 1.2 Requirements precedence to a new section 4.2 Compliance requirements. This was a request from ISO to remove requirements from the scope section.
f) Moved section 1.3 SCSI standards family into the front matter just before section 1. This was a request from ISO;
g) Added in a more specific set of rules to ordered and unordered lists in section 3.4 Editorial conventions;
h) Added the statement “notes are numbered consecutively throughout this standard” in section 3.4 Editorial conventions;
i) Moved the annex acronyms and bibliography from annex A to a new Annex to conform to ISO styles;
j) Created a real table for the command management events in section 8.4; and
k) Removed the statement “in previous versions of this standard” and replaced that wording with a reference to the actual standard. In all cases that was either SAM-2 or SAM-3. Added SAM-2 and SAM-3 to the acronyms and bibliography.
l) 09-017r1 - QUERY TASK TMF Progress Indicator [Knight].

R.4 Revision SAM-5r03 (18 November 2009)

The following T10 approved proposals were incorporated in SAM-5 in this revision:

a) 09-373r0 - Help for those who cannot find initiator/target ports in SAM-5 [Weber]; and
b) 09-381r2 - Adding a Copy Manager class into SAM-5 [Penokie].

R.5 Revision SAM-5r04 (20 January 2010)

The following T10 approved proposals were incorporated in SAM-5 in this revision:

a) 09-241r6 - SCSI Referrals [Penokie]

Added in as much of the UAS name and identifier information into annex A as is known at this time.

R.6 Revision SAM-5r05 (19 May 2010)

The following T10 approved proposals were incorporated in SAM-5 in this revision:

a) 10-163r1 - SAM5: Aborted command reporting ambiguity [Martin].

The following comment from the SPL letter ballot was accepted with fixes to Annex B and Annex D:

Comments for the SPL editor to forward to the SAM-5 editor to keep SAM-5 in sync with SPL/SAS-2.1:

a) Update all SAM-5 references to SAS-2 to SAS-2.1 and/or SPL (probably mostly SPL).
b) Annex A has been updated in sam5r04, but annex B and D have not.
c) Annex B still uses “SAS-2” as a column header.
d) Annex D still refers to “SAS-2 SSP”.

R.7 Revision SAM-5r06 (17 January 2011)

The following T10 approved proposals were incorporated in SAM-5 in this revision:

a) 10-275r0 - SAM-5 Task Management Function's In's and Out's [Weber];
b) 10-324r1 - Give Output Parameters a chance in SAM-5 [Weber];
c) 10-325r6 - SAM-5 and FCP-4: Management Logical Units [Black and Martin];
d) 10-373r0 - SAM-5: FCP-4 changes to Annex A [Peterson]; and
e) 10-384r1 - SAM-5: Issues with 10-325r6 [Penokie].

There was a missed task to command change that was accepted as part of the letter ballot resolutions for SAM-4 that was fixed based on the following e-mail received from Bill Martin on 05/17/2010:

“In SAM-5, 4.6.23 under the TASK Set class, the following list refers to a Task class. I believe that this should be a Command class, which would be consistent with the cross reference. There is no other instance of a Task class in the entire standard.”

As a result of the following e-mail from Rob Elliott received on 05/26/2010 figure 20 was fixed:

“In sam5r05, figure 20 “Eight byte LUN structure adjustments” appears to have been broken by a figure conversion from FrameMaker to Visio. See sam5r03 or earlier for how it should look.”

The following email received from Fred Knight on 12/15/2010:

“4.7.5 Single level LUN structure

Byte 4 through byte 11 in an 12-byte single level LUN structure using the extended flat space addressing method shall contain 00h (see table 13).

A 12 byte LUN structure? All the tables (including table 13) show only 8 byte structures; and table 13 shows byte 4 through byte 7 in an 8-byte single level LUN structure containing the value 00h. So the text and the table aren't consistent.

Am I missing something here?”

This is an error that came about in the resolution of SAM-4 letter ballot comments were one letter ballot comment was made based on an incorrectly byte numbered table. The numbering was corrected as the result of a different letter ballot comment, but the wording of first letter ballot comment was not adjusted when the byte numbers where corrected.

R.8 Revision SAM-5r07 (12 May 2011)

The following T10 approved proposals were incorporated in SAM-5 in this revision:

a) 11-084r2 - Extended Logical Unit Addressing flexibility [Black];
b) 11-115r1 - SPL-2 / SAM-5: Logical unit actions for Power Loss Expected;
c) 11-193r0 - SAM-5 - Define status qualifier for CHECK CONDITION status; and
d) From an email received from Fred Knight on 2/15/2011 - From section 4.7.7 (Peripheral device addressing method): If the peripheral device addressing method (see table 18) is selected, the SCSI target device should relay the received command or task management function to the addressed dependent logical unit.

   <and>

   From section 4.7.9 (Logical unit addressing method): If the logical unit addressing method (see table 20) is selected, the SCSI <<target>> device should relay the received command or task management function to the addressed dependent logical unit.

   It seems like they should both use the same language - SCSI target device - “target” should be added to 4.7.9

   And from 4.7.6:

   If the LUN specifies that the command or task management function is to be relayed to the next level (i.e. peripheral device addressing method (see 4.7.7) is selected in byte 0 and byte 1 of the eight bye LUN structure and the BUS IDENTIFIER field is set to a value greater than zero), then the current level shall use byte 0 and byte 1 of the eight byte LUN structure to determine the address of the SCSI target device to which the command or task management function is to be sent. When the command << or task management function >> is sent to the SCSI target device the eight byte LUN structure that was received shall be adjusted to create a new eight byte LUN structure (see table 14 and figure 21).

   Everything else in this paragraph says command or task management function except for this one reference in the middle of the paragraph where it only says command, and it leaves out “or task management”.

R.9 Revision SAM-5r08 (22 November 2011)

The following T10 approved proposals were incorporated in SAM-5 in this revision:

a) 11-259r1 - Representing >2 byte LUN formats [Knight];
b) 11-329r0 - SAM-5 16-bit LUN correction [Elliott];
c) 11-459r2 - XCOPYv2: Upgrades & RODs [Weber]; and
d) Made the change requested in the following email received from Rob Elliott on 7/7/2011:

Five sentences in sam5r07 use “task management procedure call” rather than “task management function procedure call”, the term that would agree with the 7.1 section name.

Example in 6.3.4:

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

should be:

When a SCSI initiator port detects an I_T nexus loss, it should terminate all its outstanding Execute Command procedure calls and all its outstanding task management function procedure calls for the SCSI target port associated with the I_T nexus with a service response of SERVICE DELIVERY OR TARGET FAILURE; and

e) Fixed a the bad cross-reference pointed out in the following email from Fred Knight on 8/26/2011:

At the end of 5.9.2, there is the following paragraph:

If the SCSI transport protocol does not enforce state synchronization as described in 4.6.14, there may be a time delay between the occurrence of the ACA condition and the time at which the application client becomes aware of the condition.

BUT, I can't find anything in 4.6.14 that talks about state synchronization. 4.6.14 talks about the SCSI Target Device class.

I think that reference is wrong it should point to 4.4.2.

f) Made the change requested in the following email received from Rob Elliott on 11/17/2011:

In sam5r07, the UML class is called “SCSI Domain”, but the 3.1.35 defined term is “domain”. I only see 6 cases where “domain” is used without the “SCSI” prefix. I think 3.1.35 should be “SCSI domain” and “SCSI” should be added to those 6 places.

sam5r07 Figure 14 title forgets to capitalize d in Domain (this refers to UML)

sam5r07 Figures 11, 28, 29, 30 capitalize a bunch of terms they shouldn't: Domain, Unit, Server, Manager, Port, Delivery Subsystem, etc. The latter three already have about half the terms in lowercase; I think using the lowercase glossary terms is better for these figures.

g) Made the change requested in the following email received from Rob Elliott on 11/22/2011:

“device model” is a defined term in SAM-5:

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

My complaint about the SAM-5 definition:

3.1.64 logical unit: A class whose objects implement, or an object that implements, a device model that manages and processes commands sent by an application client. See 4.6.19.

is that a “description of a type of SCSI target device” does not “manage and process commands”.

If “that” were replaced by “and”:

3.1.64 logical unit: A class whose objects implement, or an object that implements, a device model and manages and processes commands sent by an application client. See 4.6.19.

then the subject for “manages and processes commands” would be “a class... or an object” which is reasonable.

R.10 Revision SAM-5r09 (18 January 2012)

The following T10 approved proposals were incorporated in SAM-5 in this revision:

a) 11-122r2 - Protocol specific changes for SOP [Elliott];
b) 11-518r0 - Port attributes for UAS [Elliott];
c) 12-004r3 - Eliminate SCSI port name and SCSI port identifier terms [Elliott];
d) 12-018r0 - Add NAA Locally Assigned format for SAS SSP [Elliott];
e) 12-037r0 - iSCSI name clarifications [Elliott, Black, Chadalapaka, Knight];
f) converted the glossary to ISO style; and
g) Made the indicated editorial changes requested in an email from Rob Elliott on 11/23.2011:

Page 2

T11/2122D, Fibre Channel - Single-Byte Command Code Sets - 4 Mapping Protocol (FC-SB-4) <<GP - Fixed >>

Problems:

a) “col” is in italics

b) Are T11 ISO numbers predictable ahead of time? <<GP - Yes but ISO frowns on use using them until they get the document >>

Page 25 figure 12

receive

s/b

receive <<GP - Fixed >>

Page 25 figure 12

The “1” and “0..1” around Device Server are kind of floating; hard to be sure where the numbers apply <<GP - Fixed >>

Page 25 figure 12

The send/receive ROD token red line lacks number ranges, which are present in all other red lines in the figure <<GP - They were there it just wasn't obvious the number when with the association - Fixed >>

Page 31 figure

In other UML figures, the first character for the label of a red line like the P in “Puts” is lowercase <<GP - Fixed >>

Page 40 section 4.6.21

items a) and b) should start with lowercase <<GP - Wrong - Those are the names of operations which are capitalized. >>

Page 40 section 4.6.21

“that processes:”... needs plural items, but a) and b) are singular. Change “operation that creates” to “operations that create” and change “operation that converts” to “operations that convert” <<GP - Fixed >>

Page 43

Maybe “LUN structure using xxx method” should be “LUN structure using the xxx method” throughout here (add the word “the”) (also in the table titles)? <<GP - Text changed but no change to table titles >>

Page 43 section 4.7.3

Need a space after “(e.g.,” in the first item a) <<GP - Fixed >>

Page 43 section 4.7.3

fix two item “a)”s <<GP - Fixed >>

Page 46-47

fix table 14 crossing page break. <<GP - Fixed >>

It would be nice if there were a sentence between figure 21 and table 15 too. <<GP - No change>>

Page 55 table 27

Suggest using horizontal lines so the 2h, Eh, and Fh cells can be straddled <<GP - Gladly Fixed >>

Page 55 table 27

“-” is not supposed to be used for ranges <<GP - Fixed >>

Page 86

Note there is no COPY OPERATION ABORT command defined in any of the references yet. 11-459r2 proposes one. Might be appropriate to carry an editor's note until it shows up. << GP - If that is still true when the next revision is posted then I will do that >>

R.11 Revision SAM-5r10 (14 March 2012)

The following T10 approved proposals were incorporated in SAM-5 in this revision:

a) 11-437r4 (Triggering I_T nexus loss on other I_T nexuses) [Elliott];
b) 12-120r1 (New LUN hierarchy format) [Knight, Black, Rajagopal];
c) Made the indicated editorial changes requested in an email from Rob Elliott on 01/18.2012;

   Annex A mixes the concepts of protocol names with standard names. The tables should use the protocol name as the column header, and include the standard name defining that as a reference. In some cases, they're identical, but SAS SSP is an example where the standard defining it (SPL-n) has a different name.

   In table A.3, for example, I think the column headers should be:

   FCP (see FCP-4)

   SRP (see SRP)

   iSCSI (see iSCSI)

   SBP (see SBP-3)

   SAS SSP (see SPL-2)

   ADT (see ADT-2)

   UAS (see UAS-2)

d) Made the indicated editorial changes requested in an email from Fred Knight on 2/9/12:

   4.7.9 Logical unit addressing method - Table 20 shows the following fields: ADDRESS METHOD, TARGET, BUS NUMBER, and LUN. The text talks about the BUS NUMBER field. Figure 22 (Logical unit selection using the logical unit addressing format) however, references the BUS IDENTIFIER field in three places. It looks like the use of BUS IDENTIFIER field in figure 22 should be BUS NUMBER field. Note the usage of BUS IDENTIFIER in the item b of the list is correct.

e) Made the indicated editorial changes requested in an email from Fred Knight on 2/9/12:

   It looks like the “… following a flat space …” should actually be “following an extended flat space …”

4.7.12 Extended flat space addressing method

The extended flat space addressing method (see table 29) specifies a logical unit at the current level. The contents of all hierarchical structure addressing fields following an extended flat space addressing method addressing field shall be ignored; and

f) converted the multi-byte fields in tables to the new format.

R.12 Revision SAM-5r11 (16 May 2012)

The following T10 approved proposals were incorporated in SAM-5 in this revision:

a) 11-517r3 - SCSI port identifiers for … [Elliott];
b) 12-188r0 - Display of new long format LUN addresses [Knight];
c) 12-190r0 - Remove SCSI port is synonymous with port [Elliott];
d) 12-219r0 - Transport LUNs vs Command Parameter LUNs in SAM-5 [Weber];
e) 12-228r0 - Add LUN size to Port attributes annex [Elliott];
f) 12-256r1 - Correct Maximum ADT CDB Length [Suhler];
g) Made the indicated editorial changes requested in an email from Rob Elliott on 03/15/2012;

Page i: right-justified text is at 6.5", not 6.7"

Page i and all other pages: Missing comma in March 14 2012 (use 14 March 2012 format to avoid commas)

In 4.7.5 table 12 and elsewhere: I thought FLAT SPACE LUN in table 12 was supposed to change to FLAT SPACE ADDRESS to match tables 13 and 14. Tables 20, 30 and 32 use EXTENDED FLAT SPACE LUN and LONG EXTENDED FLAT SPACE LUN, so are inconsistent with tables 13 and 14. Tables 12, 13, 14, 20, 30, and 32 need to be consistent.

Table 13 needs the ugly … row between 1 and 3

In 4.7.5 page 49 new item b): 16 777 216 logical units s/b 16 777 216 of the logical units

In figure 20, the 01234567 are supposed to line up with the ABCDEFGH and each is supposed to be inside a box

Figure 20 might be better using STUVWXYZ than ABCDEFGH, since the latter look like hex digits (except for the G and H).

In figures 21 and 22: BUS IDENTIFIER field s/b the BUS IDENTIFIER field (several times in each)

Table 31 title needs to start with “Long”

In figures 27, 28, and 29, there is something wrong with “Logical” in the left side. The text is cropped at the top.

In 6.3.4: c) needs “an” at the beginning and c) needs to end in; not.

In tables A.3, A.4, A.5,: SPL SSP s/b SAS SSP

In table A.5, delete period from “Reported in the Device Identification VPD page (see SPC-4).”

In table B.2, SPL s/b SAS SSP

h) Made the indicated editorial changes requested in an email from Rob Elliott on 03/23/2012;
   A) These sections have parallel content, but their names are inconsistent:
      5.4 SCSI transport protocol services in support of Execute Command
      7.12 Task management SCSI transport protocol services
      6.4 Event notification SCSI transport protocol services
      Suggestions:
      5.4 SCSI transport protocol services for SCSI commands
      7.12 SCSI transport protocol services for task management functions
      6.4 SCSI transport protocol services for event notification
   B) When that's resolved:
      5.4.1 Overview

s/b
5.4.1 <5.4's name> overview
like 7.12.1.

C) Adjust 7.12.1's name if 7.12 changes.

i) Made the indicated editorial changes requested in an email from Rob Elliott on 04/27/2012;

In table 41 (pg 89) footnote c, there is a run-on sentence:

If the TST field is set to 001b (i.e., per I_T nexus) in the Control mode page (see SPC-4), there is one task set per I_T nexus, as a result, no other I_T nexuses are affected and CLEAR TASK SET is equivalent to ABORT TASK SET.

, as a

s/b

. As a

j) Made the indicated editorial changes requested in an email from Rob Elliott on 04/27/2012;

Table 46: ACA command present in the Task Set

s/b

… task set

Global:

with a CHECK CONDITION status

delete “a”

3.1.40 faulting command: command that has terminated with a status of CHECK CONDITION that resulted in the establishment of an ACA (see 3.1.8)

s/b

… terminated with CHECK CONDITION status and resulted…

Page 87

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

s/b

…with GOOD status…

Page 91

When a logical unit completes one or more commands received on an I_T nexus with a status of TASK ABORTED, the logical unit should complete all of the affected commands before entering any other commands received on that I_T nexus into the task set.

s/b

…with TASK ABORTED status…

Page 121

Assuming in each case the command completes with a GOOD status at time C, the state observed by the application client for case 1 shall be indistinguishable from the state observed for case 2.

s/b

…with GOOD status…

k) Made the indicated editorial changes requested in an email from Fred Knight 05/01/2012;

Page 59 - table 28 - Logical unit extended addressing

The new row for “long extended flat space addressing” should be “10b” not “10h”.

Check use of “well known logical unit” vs. “well-known logical unit” The - should be removed.

l) Made the indicated correction requested in an email from Rob Elliott 05/14/2012:

Bit got mistakenly upgraded to byte by the conversion from task tag -> task identifier -> command identifier: This occurred as a result of an incorrect editing change in proposal 07-157r4.

sam4r12:

3.1.120 task tag: An attribute of a task class containing up to 64 bits that uniquely identifies each task for a given I_T_L nexus (see 3.1.49) in a task set (see 3.1.126). See 4.5.23.2.

4.5.23.2

A task tag is composed of up to 64 bits.

sam4r13:

Each SCSI transport protocol defines the size of the task identifier, up to a maximum of 64 bytes, to be used by SCSI ports that support that SCSI transport protocol.

sam4r14:

4.8.2 Command identifier

… Each SCSI transport protocol defines the size of the command identifier, up to a maximum of 64 bytes, to be used by SCSI ports that support that SCSI transport protocol.

R.13 Revision SAM-5r12 (18 October 2012)

The following T10 approved proposals were incorporated in SAM-5 in this revision:

a) 12-294r2 - Complete the Definition of Incorrect Logical Unit Numbers in SAM-5 [Weber];
b) 12-029r5 - Command and Task Management Function architecture model [Elliott]
c) Made the indicated editorial changes requested in an email from Rob Elliott on 05/17/2012;

1. In table A.2

logical unit name optional (well known logical unit) is incorrect, as logical unit names are prohibited for well known logical units. (Changed optional to none).

3. In table B.2, the title: SCSI Initiator Port attributes and SCSI Target Port attributes that are supported by SOP, SPL, SRP, and UAS SCSI transport protocols

SPL s/b SAS SSP and the column header SAS s/b SAS SSP

4. In annex D:

SPL SSP: SPL Serial SCSI Protocol (see SPL). s/b SAS SSP: SAS Serial SCSI Protocol (see SPL-2).

5. In 4.7.3 pg 47

The LUN identifier defined be a transport protocol standard

be s/b by

6. In 4.7.3 pg 47:

All LUN identifiers defined by command standards shall contain 64 bits. When a 16 bit transport protocol LUN identifier is contained in a 64 bit LUN identifier defined by a command standard:

a) the 16 bits of the LUN identifier defined by the transport protocol standard shall be placed in the high order 16 bits of the 64 bit LUN identifier; and

b) all other bits of the 64 bit LUN identifier shall be set to zero.

“16 bit transport protocol LUN” s/b “16-bit transport protocol LUN”

“64 bit LUN” s/b “64-bit LUN”

[use the dash when the bit count is an adjective]

8. In 3.1.xx:

3.1.18 command descriptor block (CDB) (5.2 and SPC-4)

3.1.113 sense data (5.13 and SPC-4)

those are missing the word “see”

9. In 3.2, several of the acronyms like these (not all are shown here):

ADC-2 Automation/Drive Interface - Commands - 2 (see 2)

ADC-2 Automation/Drive Interface - Commands - 2 (see 2)

ADT-2 Automation/Drive Interface Transport Protocol - 2 (see 2)

FCP-4 SCSI Fibre Channel Protocol - 4 (see 2)

FC-SB-2 Fibre Channel - Single-Byte Command Code Sets - 4 Mapping Protocol (see 2) are missing the word “clause”

10. On pg 48-49, table 12 wraps pages without “(part 1 of 2)” in the title

11. In table 42 (pg 91), the a)b) lists are too far to the left. Should be the same as the rest of the text.

13. Telephone number formats are inconsistent on page i “507-” vs page ii “(719) “

d) Made the indicated editorial changes requested in an email from Rob Elliott on 01/26/2012;

In sam5r09 section 8.7 on page 123:

“However, processing of a collection of commands with different task priorities should cause the subset of commands with the higher task priorities to complete with status sooner in aggregate than the same subset would if the same collection of commands were submitted under the same conditions but with all task priorities being equal.”

should use “command priorities”. These was probably missed by doing a global search on “task priority” and not catching the plural;

e) added color to the constraint boxes in the UML diagrams per the conventions section;
f) Made the indicated editorial changes requested in an email from Rob Elliott on 07/12/2012:

Section 3.6.2 Notation for procedure calls (page 21) does not exactly describe the notation used in the standard for Execute Command in 5.1 (page 75) and <Function name> in 7.1 (page 110).

The following is an example of a procedure

call specification:

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

[the above line is always indented in an invisible table]

Where:

[“Where” is not used]

Found = Flag

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

[The return value is described after the Output arguments, not before the Input arguments, after a sentence like “One of the following SCSI transport protocol specific service responses shall be returned:”]

Input Arguments:

Pattern = … /* Definition of Pattern argument */

Argument containing the search pattern.

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

Contains the items to be searched for a match.

[format in an invisible table, with: rather than = and the description starting on the same line.]

Output Arguments:

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

This argument is only returned if the search succeeds.

[format in an invisible table, with: rather than = and the description starting on the same line.];

g) Made the indicated editorial changes requested in an email from Rob Elliott on 07/12/2012:

On page 109

Argument description

Argument descriptions:

Argument descriptions:

s/b

Input argument:

Input argument:

Input argument:

h) Made the indicated editorial changes requested in an email from Rob Elliott on 07/12/2012:

In sam5r11 table 41 (page 90)

ABORT TASK	6.3.1 and 7.2
ABORT TASK SET	6.3.2 and 7.3

should be:

ABORT TASK	7.2
ABORT TASK SET	7.3

06-026r4 just had the 7.x references; sam4r12 ended up incorporating it with the 6.3.x references as well.

i) Made the indicated editorial changes requested in an email from Rod Elliott on 08/08/2012; and

Section 4.6 is inconsistent on whether initiator or target is shown on the left in figures, and whether sections describing the figures are sorted using a breadth-first or depth-first scan.

Figure 11 and 15 show Target classes on the left and Initiator classes on the right (most common)

Figure 14 shows SCSI Initiator Device on the left and SCSI Target Device on the right

That differs from most other figures
They should be swapped

Within the SCSI Port range 4.6.5 to 4.6.8, Task Router (which is contained in SCSI Target Port) is after Initiator Port.

It should be after SCSI Target Port to follow depth-first sorting.

4.6.9 through 4.6.13 describe SCSI Initiator Device before 4.6.14 to 4.6.27 describe SCSI Target Device.

Those regions should be swapped to match corrected figure 14.

j) Made the indicated editorial changes requested in an email from Fred Knight on 09/18/2012:

I believe this is another lost remnant of the LINKED COMMAND removal (therefore editorial).

8.5.2 Enabled command state

A command in the enabled command state may become a current command and may complete at any time, subject to the command completion constraints specified in the Control mode page (see SPC-4). A command that has been accepted into the task set shall not complete or become a current command unless it is in the enabled command state.

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

Commands are no longer defined by the commands they process; but by the operations (or functions) they have processed.

R.14 Revision SAM-5r13 (23 January 2013)

The following T10 approved proposals were incorporated in SAM-5 in this revision:

a) 12-335r2 - Move multiple port model from SPC-4 [Elliott];
b) 13-003r0 - Add Data Transfer Terminated operation to Task Manager class [Penokie];
c) 13-004r1 - Task management function description tweaks [Elliott];
d) 13-016r1 - Fix for Incorrect Incorporation of Multiple Proposals Modifying Same Text [Penokie]; and
e) 13-020r1 - Logical Unit Groups (aka Conglomerates) in SAM-5 and SPC-4 [Weber].

Contents

Page

Tables

Figures

Foreword

This foreword is not part of American National Standard INCITS ***-201x.

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 ***-201x, 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 Industry Council, Suite 610, 1101 K Street, NW, Washington, DC 20005.

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:

Editor’s Note 1: <<Insert INCITS member list>>

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

John B. Lohmeyer, Chair
Mark Evans, Vice-Chair
Ralph O. Weber, Secretary

Editor’s Note 2: << Insert T10 member list >>

Introduction

This standard is divided into the following clauses and annexes:

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

SCSI standards family

Figure 1 shows the relationship of this standard to the other standards and related projects in the SCSI family of standards as of the publication of this standard.

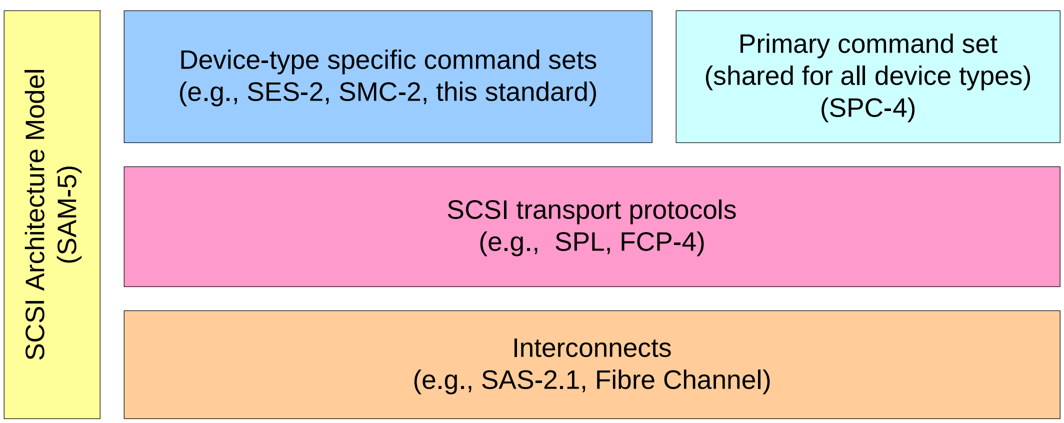


Figure 1 — SCSI document structure

The SCSI document structure in figure 1 is intended to show the general applicability of the documents to one another. Figure 1 is not intended to imply any hierarchy, protocol stack, or system architecture relationship.

The functional areas identified in figure 1 characterize the scope of standards within a group as follows:

SCSI 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 behaviors that are specific to a given device type and prescribe the requirements to be followed by a SCSI initiator device when sending commands to a SCSI target device having the specific device type. The commands and behaviors for a specific device type may include by reference commands and behaviors that are shared by all SCSI devices.

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

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

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

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

American National Standard
for Information Technology -

SCSI Architecture Model - 5 (SAM-5)

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 clause 2. This standard defines a reference model that specifies common behaviors for SCSI devices, and an abstract structure that is generic to all SCSI I/O system implementations.

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

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

a) support for the SPI-5 SCSI transport protocol;
b) Contingent Allegiance;
c) the TARGET RESET task management function;
d) basic task management model;
e) untagged tasks; and
f) linked command function.

2 Normative references

2.1 Normative references

The following standards and specifications 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:

a) approved ANSI standards;
b) approved and draft international and regional standards (ISO, IEC); and
c) approved and draft foreign standards (including JIS, and DIN).

For further information, contact ANSI Customer Service Department at 212-642-4900 (phone), 212-302-1286 (fax) or via the World Wide Web at http://www.ansi.org.

2.2 Approved references

ISO/IEC 14776-232, *Serial Bus Protocol - 3 (SBP-3)* [ANSI INCITS 375-2004]

ISO/IEC 14776-241, *SCSI RDMA Protocol (SRP)* [ANSI INCITS 365-2002]

ISO/IEC 14776-357, *Automation/Drive Interface - Commands - 2 (ADC-2)* [ANSI INCITS 441-2008]

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 14776-454, *SCSI Primary Commands - 4 (SPC-4)* [T10/1731-D]

ISO/IEC 14776-323, *SCSI Block Commands - 3 (SBC-3)* [T10/1799-D]

ISO/IEC 14776-224 *Fibre Channel Protocol for SCSI - 4 (FCP-4)* [T10/1828-D]

ISO/IEC 14776-152, *Serial Attached SCSI - 2 (SAS-2)* [T10/1760-D]

ISO/IEC 14776-192, *Automation/Drive Interface - Transport Protocol - 2 (ADT-2)* [T10/1742-D]

T11/2122D, Fibre Channel - Single-Byte Command Code Sets - 4 Mapping Protocol *(FC-SB-4)*

T10/2239D, SCSI over PCIe (SOP)

2.4 Other references

RFC 3720, *Internet Small Computer Systems Interface (iSCSI)*

NOTE 1 - Copies of IETF standards may be obtained through the Internet Engineering Task Force (IETF) at http://www.ietf.org.

ISO/IEC 19501, *Unified Modeling Language Specification Version 1.4.2*

NOTE 2 - For more information on the UML specification, contact the Object Modeling Group at http://www.omg.org.

3 Definitions, symbols, abbreviations, and conventions

3.1 Definitions

3.1.1 ACA command

command with the ACA task attribute (see 3.1.8 and 8.6.5)

3.1.2 additional sense code
combination of the ADDITIONAL SENSE CODE field and the ADDITIONAL SENSE CODE QUALIFIER field in the sense data (see 3.1.110 and SPC-3)

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

3.1.4 application client (see 4.6.27)
class whose objects are, or an object that is, the source of commands and task management function requests

3.1.5 argument
datum provided as input to or output from a procedure call (see 3.1.83)

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

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

3.1.8 auto contingent allegiance (ACA) (see 5.9)
task set condition established following the completion of a command with CHECK CONDITION status when the NACA bit is set to one in the CONTROL byte

3.1.9 background operation (see 5.5)
operation started by a command that continues processing after the command is no longer in the task set

3.1.10 blocked command state (see 8.5.3)
state in which a command is prevented from completing due to an ACA condition

3.1.11 blocking boundary (see 8.9)
task set boundary denoting a set of conditions that inhibit commands outside the boundary from entering the enabled command state

3.1.12 byte
8-bit construct

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

3.1.14 class diagram (see 3.6.1.3)
shows a set of classes and their relationships; class diagrams are used to illustrate the static design view of a system

3.1.15 client-server (see 4.4)
relationship established between a pair of distributed entities where one (the client) requests the other (the server) to perform some operation or unit of work on the client's behalf

3.1.16 client
entity that requests a service from a server; this standard defines one client, the application client

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

3.1.18 command descriptor block (CDB) (see 5.2 and SPC-4)
structure used to communicate a command from an application client to a device server with a CDB having a fixed length of 6 bytes, 10 bytes, 12 bytes, or 16 bytes, or a variable length of between 12 and 260 bytes

3.1.19 command identifier (see 4.8.2)
portion of an I_T_L_Q nexus (i.e., the Q) that is the numerical identifier of the command (see 3.1.17) within an I_T_L nexus

3.1.20 command priority (see 8.7)
relative scheduling importance of a command having the SIMPLE task attribute among the set of commands having the SIMPLE task attribute already in the task set

3.1.21 command standard (see clause 1)
SCSI standard that defines the model, commands, and parameter data for a device type (e.g., SPC-4, SBC-3)

3.1.22 completed command
command that has completed with a service response of COMMAND COMPLETE

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

3.1.24 confirmed SCSI transport protocol service (see 4.10)
service available at the SCSI transport protocol service interface that includes a confirmation of completion

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

3.1.26 copy manager (see 4.6.19)
class whose objects are, or an object that is an application client that processes third-party copy commands and manages copy operations (see SPC-4) from within a logical unit

3.1.27 current command
command that has a data transfer SCSI transport protocol service request in progress (see 5.4.3) or is in the process of sending command status; each SCSI transport protocol standard may define the SCSI transport protocol specific conditions under which a command is considered a current command

3.1.28 deferred error
error generated by a background operation (see SPC-4)

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

3.1.30 dependent logical unit (see 4.6.17.4)
logical unit that is addressed via some other logical unit(s) in a hierarchical logical unit structure (see 3.1.44)

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

3.1.32 device server (see 4.6.18)
class whose objects process, or an object that processes, commands according to the requirements for command management described in clause 8

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

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

3.1.35 dormant command state (see 8.5.4)
state in which a command is prevented from entering the enabled command state (see 3.1.36) due to the presence of certain other commands in the task set

3.1.36 enabled command state (see 8.5.2)
state in which a command may complete at any time

3.1.37 extended logical unit addressing (see 4.7.7.5)
logical unit addressing method used by special function logical units (e.g., well known logical units)

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

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

3.1.40 faulting command
command that has terminated with CHECK CONDITION status that resulted in the establishment of an ACA (see 3.1.8)

3.1.41 field
group of one or more contiguous bits, part of a larger structure (e.g., a CDB (see 3.1.18) or sense data (see 3.1.110))

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

3.1.43 hard reset
condition resulting from a power on condition or a reset event in which the SCSI device performs the hard reset operations described in 6.3.2, SPC-4, and the appropriate command standards

3.1.44 hierarchical logical unit
inverted tree structure for forming and parsing LUNs (see 3.1.65) containing up to four addressable levels (see 4.6.13)

3.1.45 I_T nexus (see 4.8)
nexus between a SCSI initiator port and a SCSI target port

3.1.46 I_T nexus loss
condition resulting from a hard reset condition or an I_T nexus loss event in which the SCSI device performs the I_T nexus loss operations described in 6.3.4, SPC-4, and the appropriate command standards

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

3.1.48 I_T_L nexus (see 4.8)
nexus between a SCSI initiator port, a SCSI target port, and a logical unit

3.1.49 I_T_L_Q nexus (see 4.8)
nexus between a SCSI initiator port, a SCSI target port, a logical unit, and a command

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

3.1.51 I_T_L_x nexus (see 4.8)
either an I_T_L nexus or an I_T_L_Q nexus

3.1.52 I/O operation
operation defined by a command or a task management function

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

3.1.54 implicit head of queue (see 8.2)
optional processing model for specified commands wherein a command may be treated as if it had been received with a HEAD OF QUEUE task attribute

3.1.55 incorrect logical unit number (see 4.7.1)
logical unit number of a logical unit that does not exist in the SCSI target device when addressed through a given I_T nexus

3.1.56 initiator port identifier (see 4.6.8.2)
value by which a SCSI initiator port is referenced within a SCSI domain

3.1.57 initiator port name (see 4.6.8.3)
name (see 3.1.69) of a SCSI initiator port that is world wide unique within the SCSI transport protocol of the SCSI domain of that SCSI initiator port ~~(see 4.6.5)~~; the name may be made available to other SCSI devices or SCSI ports in that SCSI domain in SCSI transport protocol specific ways

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

3.1.59 in transit
information that has been delivered to a service delivery subsystem for transmission, but not yet arrived at the intended recipient

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

3.1.61 link
individual connection between two objects in an object diagram representing an instance of an association

3.1.62 logical unit (see 4.6.17)
class whose objects implement, or an object that implements, a device model that manages and processes commands sent by an application client

3.1.63 logical unit inventory
list of the LUNs reported by a REPORT LUNS command (see SPC-4)

3.1.64 logical unit name
name (see 3.1.69) of a logical unit that is world wide unique within the SCSI transport protocol of a SCSI domain in which the SCSI device containing the logical unit has SCSI ports (see 4.6.4.2); the logical unit name may be made available to other SCSI devices or SCSI ports in SCSI transport protocol specific ways

3.1.65 logical unit number (LUN) (see 4.7)
for transport protocol standards, 64-bit or 16-bit identifier for a logical unit; for command standards, 64-bit identifier for a logical unit

3.1.66 logical unit reset
condition resulting from a hard reset condition or a logical unit reset event in which the logical unit performs the logical unit reset operations described in 6.3.3, SPC-4, and the appropriate command standards

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

3.1.68 multiplicity
when referring to classes (see 3.1.13), an indication of the range of allowable instances that a class or an attribute may have

3.1.69 name
label of an object that is unique within a specified context and should never change

3.1.70 nexus (see 4.8)
relationship between two SCSI devices, and the SCSI initiator port and SCSI target port objects within those SCSI devices

3.1.71 non-faulted I_T nexus
I_T nexus that is not a faulted I_T nexus (see 3.1.38)

3.1.72 object
entity with a well-defined boundary and identity that encapsulates state and behavior all objects are instances of classes (see 3.1.58)

3.1.73 management logical unit (see 4.6.24)
class whose objects are each, or an object that is, a logical unit that only performs management functions; management logical units allow an application client to issue requests to receive specific information and manage specific information relating to a SCSI target device

3.1.74 object diagram (see 3.6.1.4)
shows a set of objects and their relationships at a point in time. Object diagrams are used to illustrate static shapshots of instances of the things found in class diagrams

3.1.75 operation
service that may be requested from any object of the class in order to affect behavior; operations describe what a class is allowed to do and may be a request or a query; a request may change the state of the object but a query should not

3.1.76 peer entities
entities within the same layer (see 3.1.60)

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

3.1.78 power loss expected
condition resulting from a power loss expected event in which the logical unit performs the power loss expected operations described in 6.3.5, SPC-4, and the appropriate transport protocol and command standards

3.1.79 power loss expected event
event that results in a power loss expected condition (see 3.1.78) as described in 6.3.5

3.1.80 power on
condition resulting from a power on event in which the SCSI device performs the power on operations described in 6.3.1, SPC-4, and the appropriate command standards

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

3.1.82 procedure
operation that is invoked through an external calling interface

3.1.83 procedure call (see 3.6.2)
model used by this standard for the interfaces involving both the SAL (see 3.1.92) and STPL (see 3.1.103), having the appearance of a programming language function call

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

3.1.85 queue
arrangement of commands within a task set (see 3.1.129)

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

3.1.87 reference model
standard model used to specify system requirements in an implementation-independent manner

3.1.88 relative port identifier (see 4.6.5.2)
identifier for a SCSI port that is unique within a SCSI device

3.1.89 request-response transaction
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

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

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

3.1.92 SCSI application layer (SAL)
protocols and procedures that implement or issue commands and task management functions by using services provided by a STPL (see 3.1.103)

3.1.93 SCSI device (see 4.6.4)
class whose objects are, or an object that is, connected to a service delivery subsystem and supports a SCSI application protocol

3.1.94 SCSI device name
name (see 3.1.69) 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 4.6.4.2); the SCSI device name may be made available to other SCSI devices or SCSI ports in SCSI transport protocol specific ways

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

3.1.96 SCSI event (see clause 6)

condition defined by this standard (e.g., logical unit reset) that is detected by SCSI device and that requires notification of its occurrence within the SCSI device

3.1.97 SCSI I/O system
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

3.1.98 SCSI initiator device
class whose objects originate, or an object that originates, device service and task management requests to be processed by a SCSI target device and receives device service and task management responses from SCSI target devices

3.1.99 SCSI initiator port (see 4.6.8)
class whose objects act, or an object that acts, as the connection between application clients and a service delivery subsystem through which server requests and server responses are routed

3.1.100 SCSI port (see 4.6.5)
class whose objects connect, or an object that connects, the application client, device server or task manager to a service delivery subsystem through which server requests and server responses are routed; a SCSI port is one of: a SCSI initiator port (see 3.1.99) or a SCSI target port (see 3.1.102)

3.1.101 SCSI target device (see 4.6.9)
class whose objects receive, or an object that receives, device service and task management requests for processing and sends device service and task management responses to SCSI initiator devices

3.1.102 SCSI target port (see 4.6.6)
class whose objects act, or an object that acts, as the connection between device servers and task managers and a service delivery subsystem through which server requests and server responses are routed

3.1.103 SCSI transport protocol layer (STPL)
protocol and services used by a SAL (see 3.1.92) to transport data representing a SCSI application protocol transaction

3.1.104 SCSI transport protocol service confirmation
procedure call from the STPL notifying the SAL that a SCSI transport protocol service request has completed

3.1.105 SCSI transport protocol service indication
procedure call from the STPL notifying the SAL that a SCSI transport protocol transaction has occurred

3.1.106 SCSI transport protocol service request
procedure call to the STPL to begin a SCSI transport protocol service transaction

3.1.107 SCSI transport protocol service response
procedure call to the STPL containing a reply from the SAL in response to a SCSI transport protocol service indication

3.1.108 SCSI transport protocol specific
implementation of the referenced item is defined by a SCSI transport protocol standard (see 2)

3.1.109 sender
client or server that originates a service delivery transaction

3.1.110 sense data (see 5.13 and SPC-4)
data describing an error or exceptional condition that a device server delivers to an application client in the same I_T_L_Q nexus transaction as the status or as parameter data in response to a REQUEST SENSE command (see SPC-4); fields in the sense data are referenced by name in this standard

3.1.111 sense key
SENSE KEY field in the sense data (see 3.1.110 and SPC-4)

3.1.112 server
entity that performs a service on behalf of a client

3.1.113 server request
transaction from a client to a server invoking a service

3.1.114 server response
transaction from a server to a client conveying the result of a request

3.1.115 service
any operation or function performed by a SCSI object that is invoked by other SCSI objects

3.1.116 service delivery failure
any non-recoverable error causing the corruption or loss of one or more service delivery transactions while in transit

3.1.117 service delivery subsystem (see 4.6.3)
class whose objects are, or an object that is, 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

3.1.118 service delivery transaction
request or response sent through a service delivery subsystem

3.1.119 standard INQUIRY data
data returned to an application client as a result of an INQUIRY command (see SPC-4) with the EVPD bit set to zero; fields in the standard INQUIRY data are referenced by name in this standard

3.1.120 target port identifier (see 4.6.6.2)
value by which a SCSI target port is referenced within a SCSI domain

3.1.121 target port name (see 4.6.6.3)
name (see 3.1.69) of a SCSI target port that is world wide unique within the SCSI transport protocol of the SCSI domain of that SCSI target port~~ (see 4.6.5)~~; the name may be made available to other SCSI devices or SCSI ports in that SCSI domain in SCSI transport protocol specific ways

3.1.122 task
synonymous with command (see 3.1.17 and Annex C)

3.1.123 task attribute (see 8.6)
attribute of a command (see 3.1.17) that specifies the processing relationship of a command with regard to other commands in the task set (see 3.1.129)

3.1.124 task management function (see clause 7)
task manager service capable of being requested by an application client to affect the processing of one or more commands

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

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

3.1.127 task manager (see 4.6.20)
class whose objects control, or an object that controls the sequencing of commands and processes task management functions

3.1.128 task router (see 4.6.7)
class whose objects route, or an object that routes commands and task management functions between a service delivery subsystem (see 3.1.117) and the appropriate task manager(s)

3.1.129 task set (see 4.6.21)
class whose objects are, or an object that is, a group of commands within a logical unit, whose interaction is dependent on the task management (e.g., queuing) and ACA requirements

3.1.130 task tag (see 3.1.19)
term used by previous versions of this standard (see Annex C)

3.1.131 token
representation of a collection of data

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

3.1.133 transport protocol standard
SCSI standard that defines a transport protocol (e.g., SPL)

3.1.134 unconfirmed SCSI transport protocol service (see 4.10)
service available at the SCSI transport protocol service interface that does not result in a completion confirmation

3.1.135 well known logical unit (see 4.6.25)
class whose objects are each, or an object that is, a logical unit that only performs specific functions. Well known logical units allow an application client to issue requests to receive and manage specific information relating to a SCSI target device

3.1.136 well known logical unit number (W-LUN) (see 4.7.7.5.1)
LUN that identifies a well known logical unit

3.2 Acronyms

ACA	Auto Contingent Allegiance (see 3.1.8)
ADC-2	Automation/Drive Interface - Commands - 2 (see clause 2)
ADT-2	Automation/Drive Interface Transport Protocol - 2 (see clause 2)
CDB	Command Descriptor Block (see 3.1.18)
CRN	Command Reference Number
FCP-4	SCSI Fibre Channel Protocol - 4 (see clause 2)
FC-SB-2	Fibre Channel - Single-Byte Command Code Sets - 4 Mapping Protocol (see clause 2)
iSCSI	Internet SCSI (see RFC 3720, http://www.ietf.org/rfc/rfc3720.txt)
ISO	Organization for International Standards
LUN	Logical Unit Number (see 3.1.65)
n/a	Not Applicable
RAID	Redundant Array of Independent Disks
ROD	Representation Of Data (see SPC-4)
SAL	SCSI application layer (see 3.1.92)
SAM-2	SCSI Architecture Model - 2
SAM-3	SCSI Architecture Model - 3

SAS-2	Serial Attached SCSI - 2 (see clause 2)
SBC-3	SCSI Block Commands - 3 (see clause 2)
SBP-3	Serial Bus Protocol - 3 (see clause 2)
SCSI	The architecture defined by the family of standards described in clause 2
SPC-4	SCSI Primary Commands - 3 (see clause 2)
SRP	SCSI RDMA Protocol (see clause 2)
STPL	SCSI transport protocol layer (see 3.1.103)
VPD	Vital Product Data (see SPC - 4)
W-LUN	Well known logical unit number (see 3.1.136)
UML	Unified Modeling Language

3.3 Keywords

3.3.1 invalid
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
keyword indicating an item that is required to be implemented as defined in this standard

3.3.3 may
keyword that indicates flexibility of choice with no implied preference. May is synonymous with the phrase “may or may not”

3.3.4 may not
keyword that indicates flexibility of choice with no implied preference. May not is synonymous with the phrase “may or may not”

3.3.5 obsolete
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 prohibited
keyword used to describe a feature, function, or coded value that is defined in a a non-SCSI standard (i.e., a standard that is not a member of the SCSI family of standards) to which this standard makes a normative reference where the use of said feature, function, or coded value is not allowed for implementations of this standard

3.3.8 reserved
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 an error

3.3.9 restricted
keyword referring to bits, bytes, words, and fields that are set aside for other identified standardization purposes. A restricted bit, byte, word, or field shall be treated as a reserved bit, byte, word or field in the context where the restricted designation appears

3.3.10 shall
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

3.3.11 should
keyword indicating flexibility of choice with a strongly preferred alternative. Equivalent to the phrase “it is strongly recommended”

3.3.12 vendor specific
specification of the referenced item is determined by the SCSI device vendor

3.4 Editorial conventions

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

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

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

Quantities having a defined numeric value are identified by large capital letters. CHECK CONDITION, for example, refers to the numeric quantity defined in table 40 (see 5.3.1). Quantities having a discrete but unspecified value are identified using small capital letters. As an example, COMMAND 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 lowercase or uppercase letters show no ordering relationship between the listed items.

EXAMPLE 1 - The following list shows no relationship between the named items:

a) red (i.e., one of the following colors):
   A) crimson: or
   B) amber;
b) blue; or
c) green.

Lists sequenced by numbers show an ordering relationship between the listed items.

EXAMPLE 2 - The following list shows an ordered relationship between the named items:

1) top;
2) middle; and
3) bottom.

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 and notes are numbered consecutively throughout this standard.

3.5 Numeric and character conventions

3.5.1 Numeric conventions

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

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

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

This standard uses the following conventions for representing decimal numbers:

a) the decimal separator (i.e., separating the integer and fractional portions of the number) is a period;
b) the thousands separator (i.e., separating groups of three digits in a portion of the number) is a space;
c) the thousands separator is used in both the integer portion and the fraction portion of a number; and
d) the decimal representation for a year is 1999 not 1 999.

Table 1 shows some examples of decimal numbers using various conventions.

Table 1 — Numbering conventions

French	English	This standard
0,6	0.6	0.6
3,141 592 65	3.14159265	3.141 592 65
1 000	1,000	1 000
1 323 462,95	1,323,462.95	1 323 462.95

3.5.2 Byte encoded character strings conventions

When this standard requires one or more bytes to contain specific encoded characters, the specific characters are enclosed in single quotation marks. The single quotation marks identify the start and end of the characters that are required to be encoded but are not themselves to be encoded. The characters that are to be encoded are shown in the case that is to be encoded.

An ASCII space character (i.e., 20h) may be represented in a string by the character’¬’ (e.g., ’SCSI¬device’.). The encoded characters and the single quotation marks that enclose them are preceded by text that specifies the character encoding methodology and the number of characters required to be encoded.

> EXAMPLE - Using the notation described in this subclause, stating that eleven ASCII characters ‘SCSI device’ represent encoded characters is the same as writing out the following sequence of byte values: 53h 43h 53h 49h 20h 64h 65h 76h 69h 63h 65h.

3.6 Notation conventions

3.6.1 UML notation conventions

3.6.1.1 Notation conventions overview

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

See 3.6.1.3 for the conventions used for class diagrams.

See 3.6.1.4 for the conventions used for object diagrams.

3.6.1.2 Constraint and note conventions

Class diagrams and object diagrams may include constraints, which specify requirements, and notes, which are informative.

Table 2 shows the notation used for constraints and notes.

Table 2 — Constraint and note notation

Notation	Description
{Constraint text}	The presence of the curly brackets (i.e. {}) defines constraint that is a normative requirement. An example of a constraint is shown in figure 2.
note text	The absence of curly brackets defines a note that is informative. An example of a note is shown in figure 3.

3.6.1.3 Class diagram conventions

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

Table 3 — Class diagram notation for classes

Notation	Description
Class Name **Class Name** **Class Name**	A class with no attributes or operations.
Class Name Attribute01[1] Attribute02[1] **Class Name** Attribute01[1] Attribute02[1]	A class with attributes and no operations.
Class Name Operation01() Operation02()	A class with operations and no attributes.
Class Name Attribute01[1] Attribute02[1] Operation01() Operation02()	A class with attributes and operations.
Class Name Attribute01[1..*] Attribute02[1] Operation01() Operation02()	A class with attributes that have a specified multiplicity (see table 4) and operations.

Table 4 shows the notation used to indicate multiplicity in class diagrams.

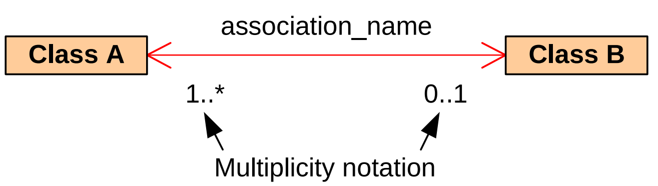
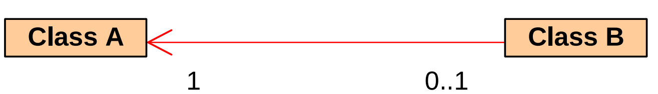
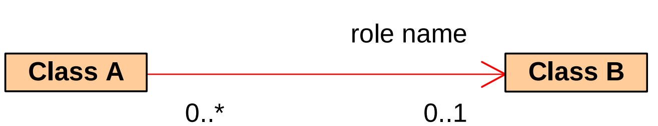
Table 4 — Multiplicity notation

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

Table 5 shows the notation used to denote association (i.e., “knows about”) relationships between classes.

Unless the two classes in an association relationship also have an aggregation relationship, association relationships have multiplicity notation (see table 4) at each end of the relationship line.

Table 5 — Class diagram notation for associations

Notation	Description
	Class A knows about Class B (i.e., read as “Class A association_name Class B”) and Class B knows about Class A (i.e., read as “Class B association name Class A”).
	Class B knows about Class A (i.e., read as “Class B knows about Class A”) but Class A does not know about Class B.
	Class A knows about Class B (i.e., read as “Class A uses the role name attribute of Class B”) but Class B does not know about Class A.
Note - The use of role names and association names are optional.	

See figure 1 for examples of association relationships between classes.

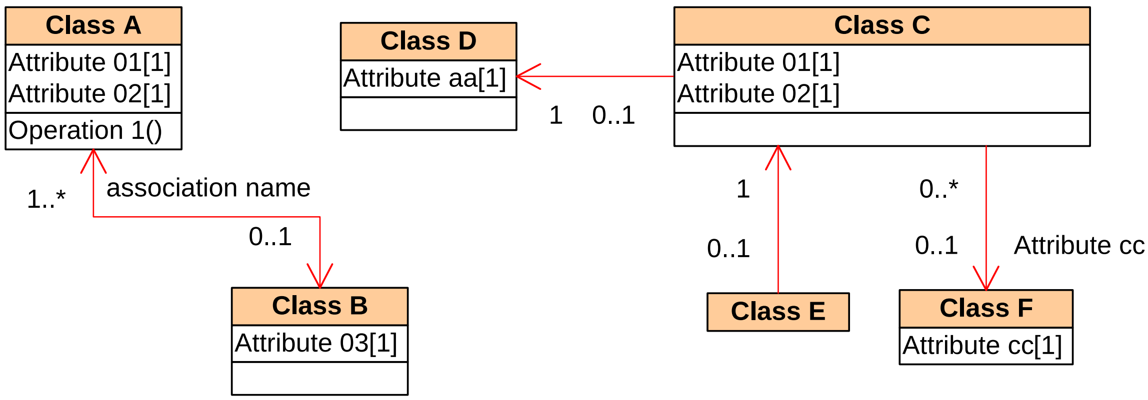
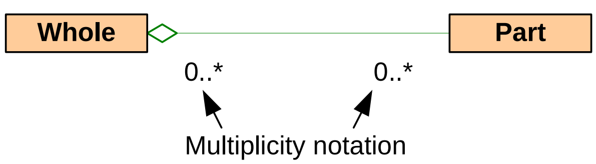


Figure 1 — Examples of association relationships in class diagrams

Table 6 shows the notation used to denote aggregation (i.e., “is a part of” or “contains”) relationships between classes. The aggregation relationship is a specific type of association and always include multiplicity notation (see table 4) at each end of the relationship line.

Table 6 — Class diagram notation for aggregations

Notation	Description
	The Part class is part of the Whole class and may continue to exist even if the Whole class is removed (i.e, read as “the whole contains the part”).
	The Part class is part of the Whole class, shall only belong to one Whole class, and shall not continue to exist if the Whole class is removed (i.e., read as “the whole contains the part”).

See figure 2 for examples of aggregation relationships between classes.

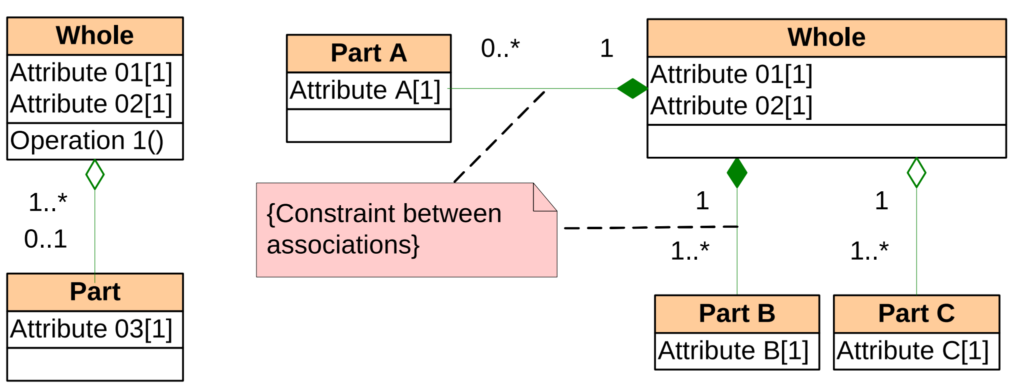
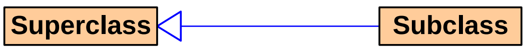


Figure 2 — Examples of aggregation relationships in class diagrams

Table 7 shows the notation used to denote generalization (i.e., “is a kind of”) relationships between classes.

Table 7 — Class diagram notation for generalizations

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

See figure 3 for examples of generalization relationships between classes.

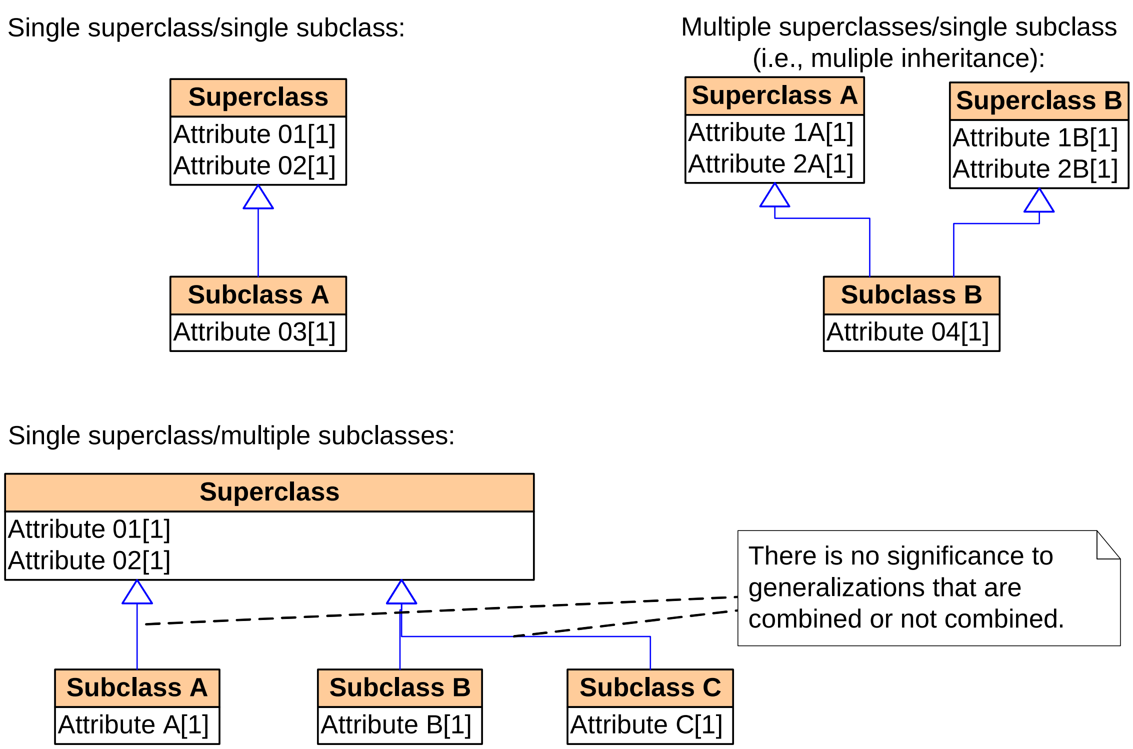


Figure 3 — Example of generalization relationships in class diagrams

Table 8 shows the notation used to denote dependency (i.e., “depends on”) relationships between classes.

Table 8 — Class diagram notation for dependency

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

See figure 4 for an example of a dependency relationship between classes.

Figure 4 — Example of a dependency relationship in class diagrams

3.6.1.4 Object diagram conventions

Table 9 shows the notation used for objects in object diagrams.

Table 9 — Object diagram notation for objects

Notation	Description
label : Class Name	Notation for a named object with no attributes.
label : Class Name Attribute01 = x Attribute02 = y	Notation for a named object with attributes.
: Class Name	Notation for an anonymous object with no attributes.
: Class Name Attribute01 = x Attribute02 = y	Notation for an anonymous object with attributes.

Table 10 shows the notation used to denote link relationships between objects.

Table 10 — Object diagram notation for link

Notation	Description
: Object A ——— : Object B	An instance of an association between object A and object B.

See figure 5 for examples of a link relationships between objects.

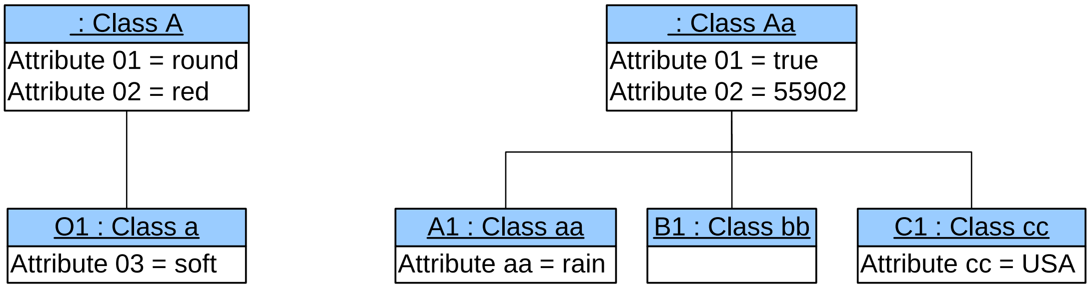


Figure 5 — Examples of link relationships for object diagrams

3.6.2 Notation for procedure calls

In this standard, the model for functional interfaces between entities is a procedure call (see 3.1.83). Such interfaces are specified using the following notation:

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

Where:

Result:A single value representing the outcome of the procedure call.

Procedure Name:A descriptive name for the function modeled by the procedure call.

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

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

“[…]”:Brackets enclosing optional or conditional arguments.

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

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

Input arguments:

Pattern: Argument containing the search pattern.

Item List: **Item<NN>** contains the items to be searched for a match.

Output arguments:

Item Found: Item located by the search procedure call. This argument is only returned if the search succeeds.

3.6.3 Notation for state diagrams

All state diagrams use the notation shown in figure 6.

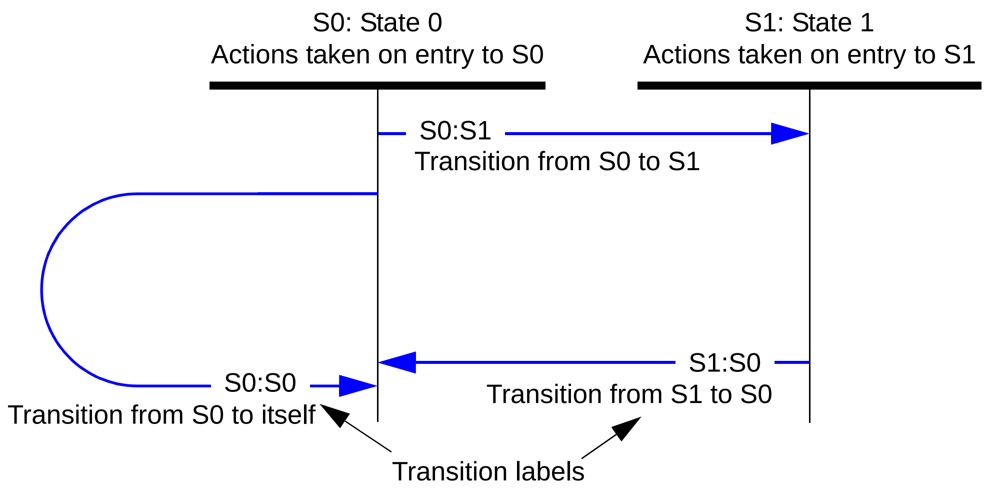


Figure 6 — Example state diagram

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 6 as an example, the transition list 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 instantaneous; and
c) every time a state is entered, the actions of that state are started. Note that this means that a transition that points back to the same state restarts the actions from the beginning.

4 SCSI architecture model

4.1 Introduction

The purpose of the SCSI architecture model is to:

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

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

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

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

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

Service interfaces are defined between distributed classes and protocol layers. The template for a distributed service interface is the client-server model described in 4.3. The structure of a SCSI I/O system is specified in 4.5 by defining the relationship among classes. The set of distributed services to be provided are specified in clause 5 and clause 7.

Requirements that apply to each SCSI transport protocol standard are specified in the SCSI transport protocol service model described in 5.4, 6.4, and 7.12. The model describes required behavior in terms of layers, classes within layers and SCSI transport protocol service transactions between layers.

4.2 Compliance requirements

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

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

As shown in figure 7, all SCSI implementation standards shall reflect the generic requirements defined herein. In addition, an implementation claiming SCSI compliance shall conform to the applicable implementation requirements defined in this standard and the appropriate SCSI implementation standards. In the event of a conflict between this document and other SCSI standards under the jurisdiction of technical committee T10, the requirements of this standard shall apply.

.

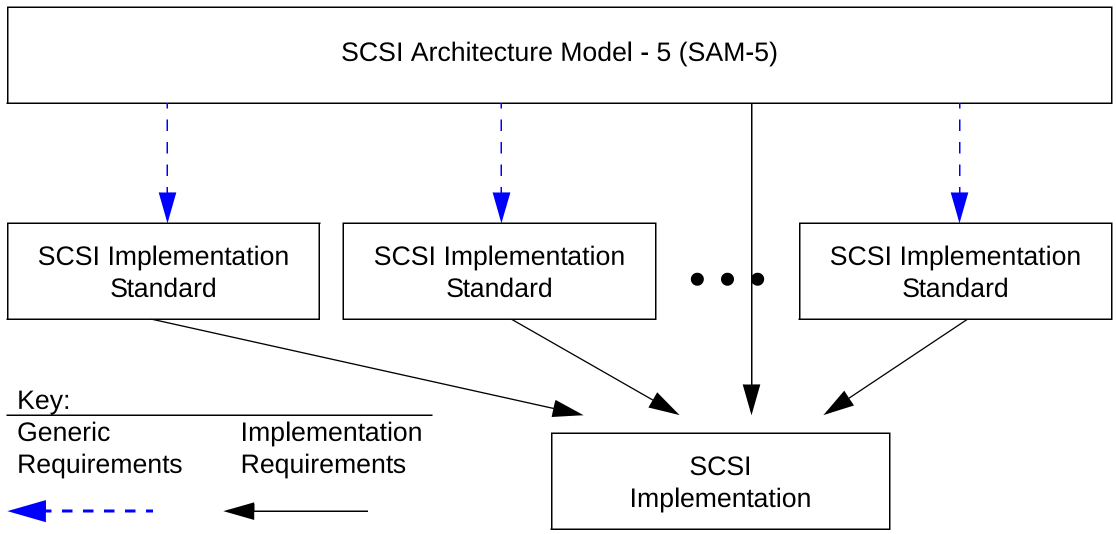


Figure 7 — Requirements precedence

4.3 The SCSI distributed service model

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

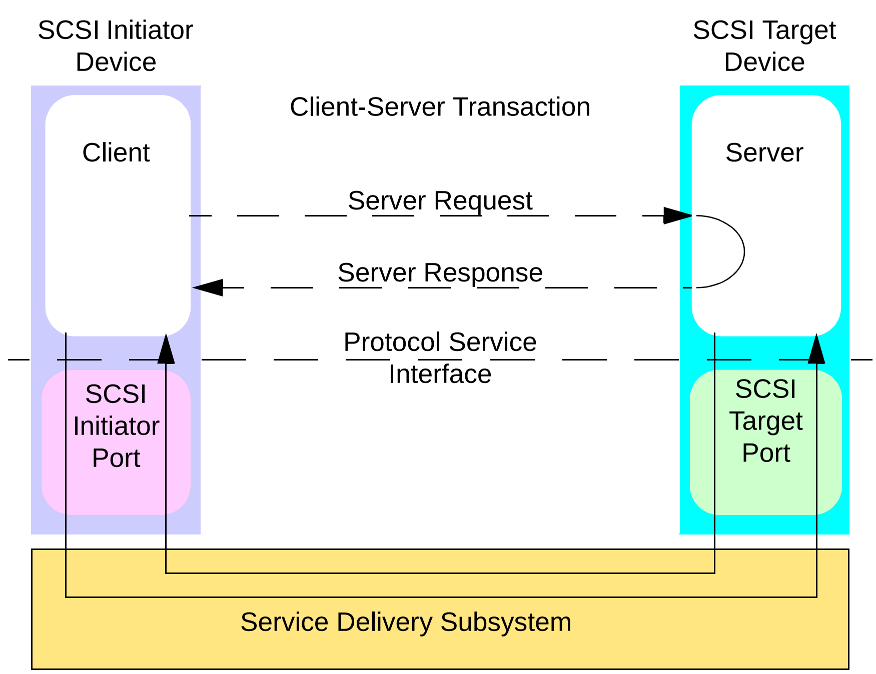


Figure 8 — Client-server model

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

As seen by the client, a server request becomes pending when it is passed to the SCSI initiator port for transmission and completes when the server response is received or when a failure notification is received. As seen by the server, the server request becomes pending upon receipt and completes when the server response is passed to the SCSI target port for return to the client. As a result there may be a time skew between the server and client's perception of server request status and server state.

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

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

4.4 The SCSI client-server model

4.4.1 SCSI client-server model overview

As shown in figure 9, each SCSI target device provides services performed by device servers and task management functions performed by task managers. A logical unit is a class that implements one of the device functional models described in the SCSI command standards and processes commands (e.g., reading from or writing to the media). Each command defines a unit of work to be performed by the logical unit that may be externally referenced and controlled through requests issued to the task manager.

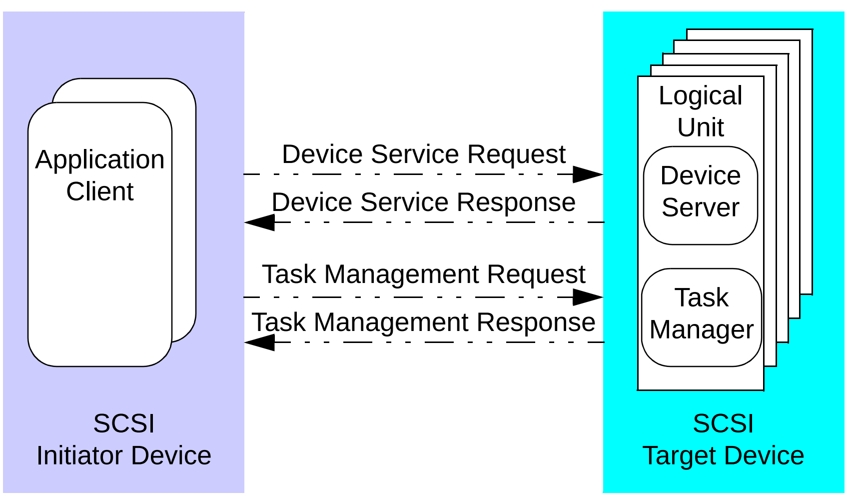


Figure 9 — SCSI client-server model

All requests originate from application clients residing within a SCSI initiator device. An application client is independent of the interconnect and SCSI transport protocol (e.g., an application client may correspond to the device driver and any other code within the operating system that is capable of managing I/O requests without requiring knowledge of the interconnect or SCSI transport protocol).

As described in 4.3, each request takes the form of a procedure call with arguments and a status to be returned. An application client may request processing of a command or a task management function through a request directed to the device server within a logical unit. Device service requests are used to request the processing of commands (see clause 5) and task management requests are used to request the processing of task management functions (see clause 7).

4.4.2 Synchronizing client and server states

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

SCSI transport protocols may require the SCSI target device to verify that the server response has been received without error 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 SCSI architecture model, there may be a time lag between the occurrence of a state change within the SCSI target device and the SCSI initiator device's awareness of that change.

This standard assumes that state synchronization, if required by a SCSI transport protocol standard, is enforced by a service delivery subsystem transparently to the server (i.e., whenever the server invokes a SCSI transport protocol service to return a response as described in 7.12 and 5.4. It is assumed that the SCSI port for such a SCSI transport protocol does not return control to the server until the server response has been delivered without error to the SCSI initiator device).

4.4.3 Server request/response ordering

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

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

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

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

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

4.5 The SCSI structural model

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

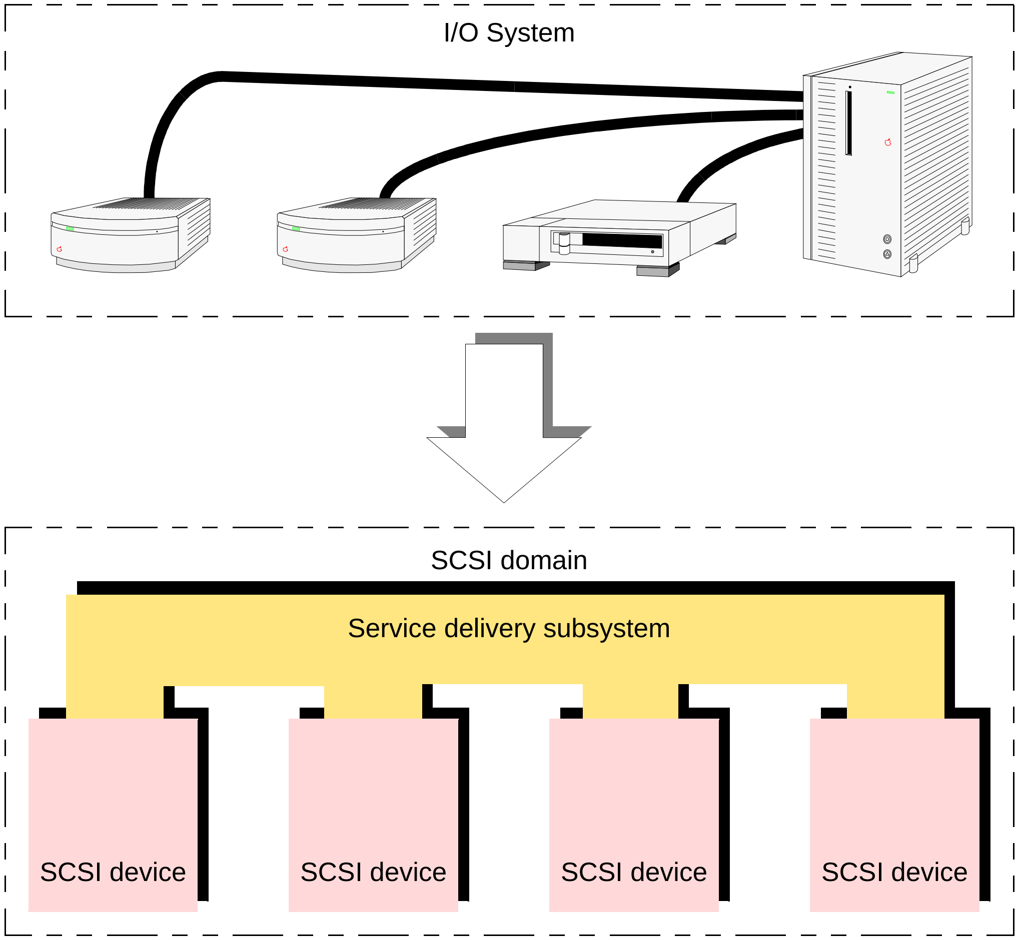


Figure 10 — SCSI I/O system and SCSI domain model

4.6 SCSI classes

4.6.1 SCSI classes overview

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

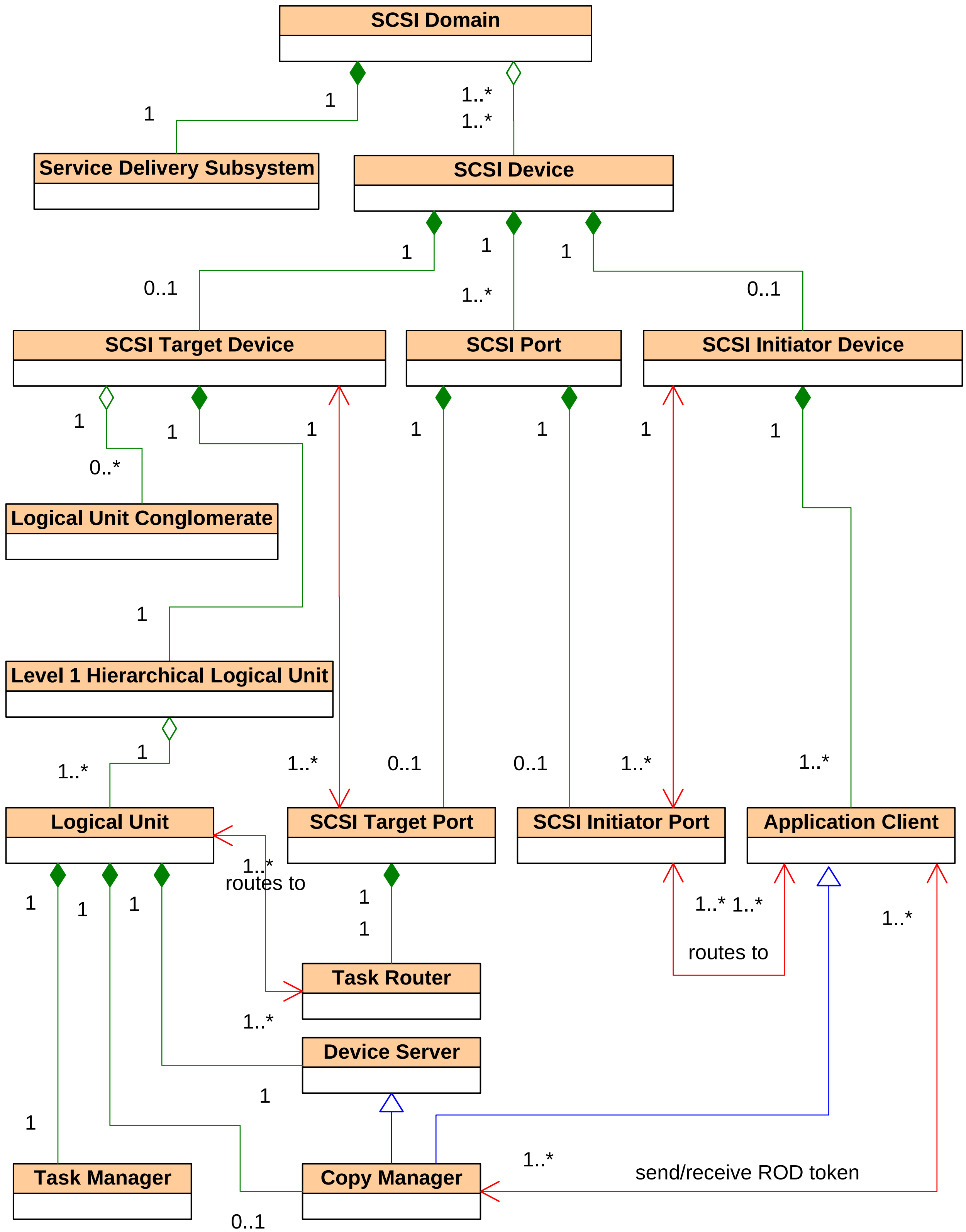


Figure 11 — SCSI Domain class diagram overview

4.6.2 SCSI Domain class

The SCSI Domain class (see figure 12) contains the:

a) Service Delivery Subsystem class (see 4.6.3); and
b) SCSI Device class (see 4.6.4).

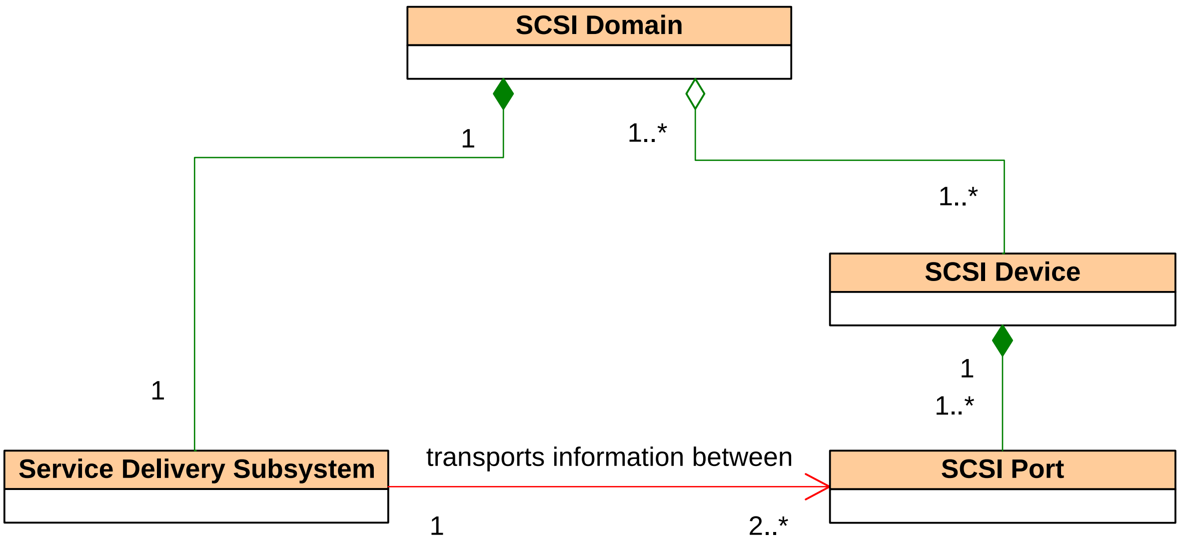


Figure 12 — SCSI Domain class diagram

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

a) one service delivery subsystem; and
b) one or more SCSI devices, each of which shall contain:
    A) one or more SCSI ports.

Instantiation requirements for the SCSI Device class are also described in 4.6.4.1.

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

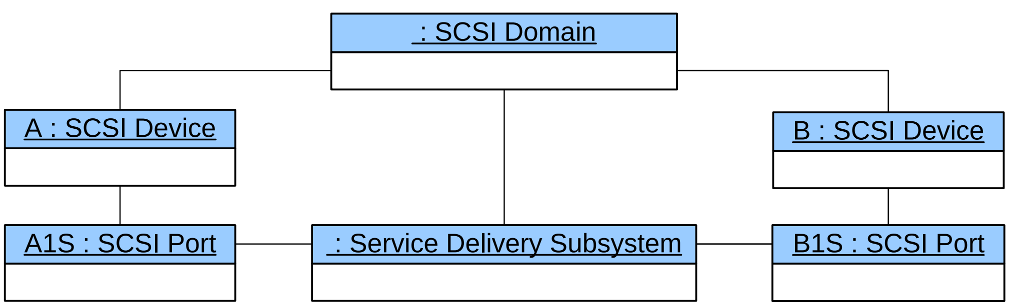


Figure 13 — SCSI Domain object diagram

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

4.6.3 Service Delivery Subsystem class

The Service Delivery Subsystem class (see figure 12) connects all the SCSI ports (see 3.1.100) in the SCSI domain, providing a mechanism through which application clients communicate with device servers and task managers.

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

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

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

4.6.4 SCSI Device class

4.6.4.1 SCSI Device class overview

The SCSI Device class (see figure 14) contains the:

a) SCSI Port class (see 4.6.5); and
b) SCSI Initiator Device class (see 4.6.26), the SCSI Target Device class (see 4.6.9), or both.

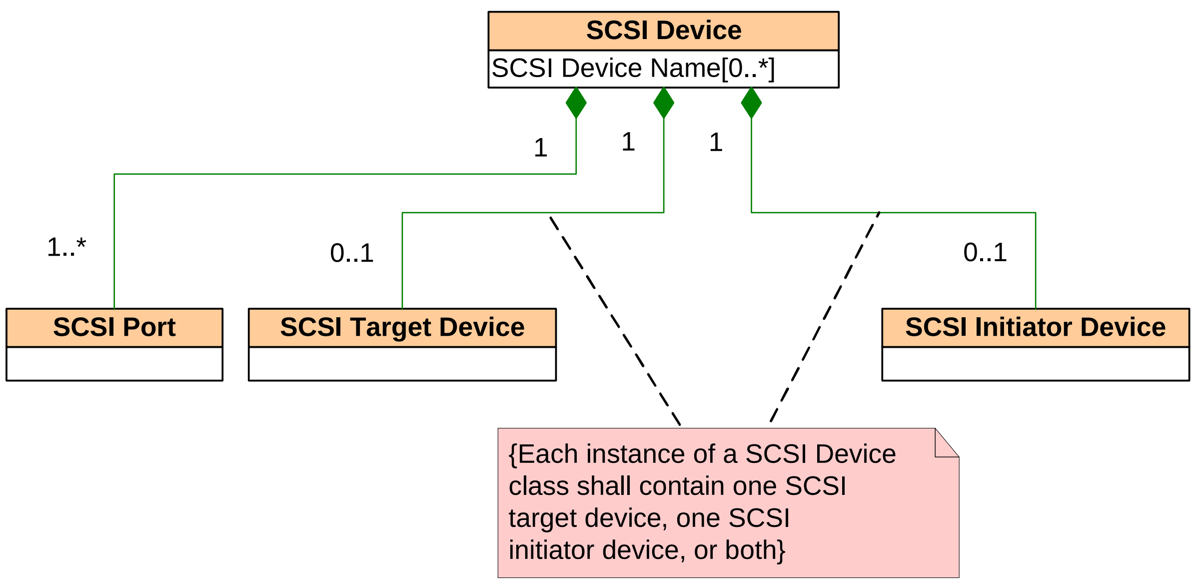


Figure 14 — SCSI Device class diagram

Each instance of a SCSI Device class shall contain:

a) one or more SCSI ports; and
b) one SCSI target device, one SCSI initiator device, or both.

Instantiation requirements for the SCSI Device class are also described in 4.6.2.

4.6.4.2 SCSI Device Name attribute

The SCSI Device Name attribute contains a name (see 3.1.69) for a SCSI device that is world wide unique within the SCSI transport protocol of each SCSI domain in which the SCSI device has SCSI ports. For each supported SCSI transport protocol, a SCSI device shall have no more than one (i.e., zero or one) SCSI Device Name attribute that is not in the SCSI name string format (see SPC-4). A SCSI device shall have no more than one (i.e., zero or one) SCSI Device Name attribute in the SCSI name string format regardless of the number of SCSI transport protocols supported by the SCSI device. If a SCSI device has a SCSI Device Name attribute in the SCSI name string format then the SCSI device should have only one SCSI Device Name attribute. A SCSI device name shall never change and may be used for persistent identification of a SCSI device in contexts where specific references to initiator port names, target port names, initiator port identifiers, or target port identifiers are not required.

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

The SCSI device name for a SCSI target device may be reported in the Device Identification VPD pages designation descriptor for SCSI target devices (see SPC-4). The SCSI device name for a SCSI initiator device is reported by methods outside the scope of this standard.

4.6.5 SCSI Port class

4.6.5.1 SCSI Port class overview

The SCSI Port class (see figure 15) contains the:

a) SCSI Target Port class (see 4.6.6);
b) SCSI Initiator Port class (see 4.6.8); or
c) both.

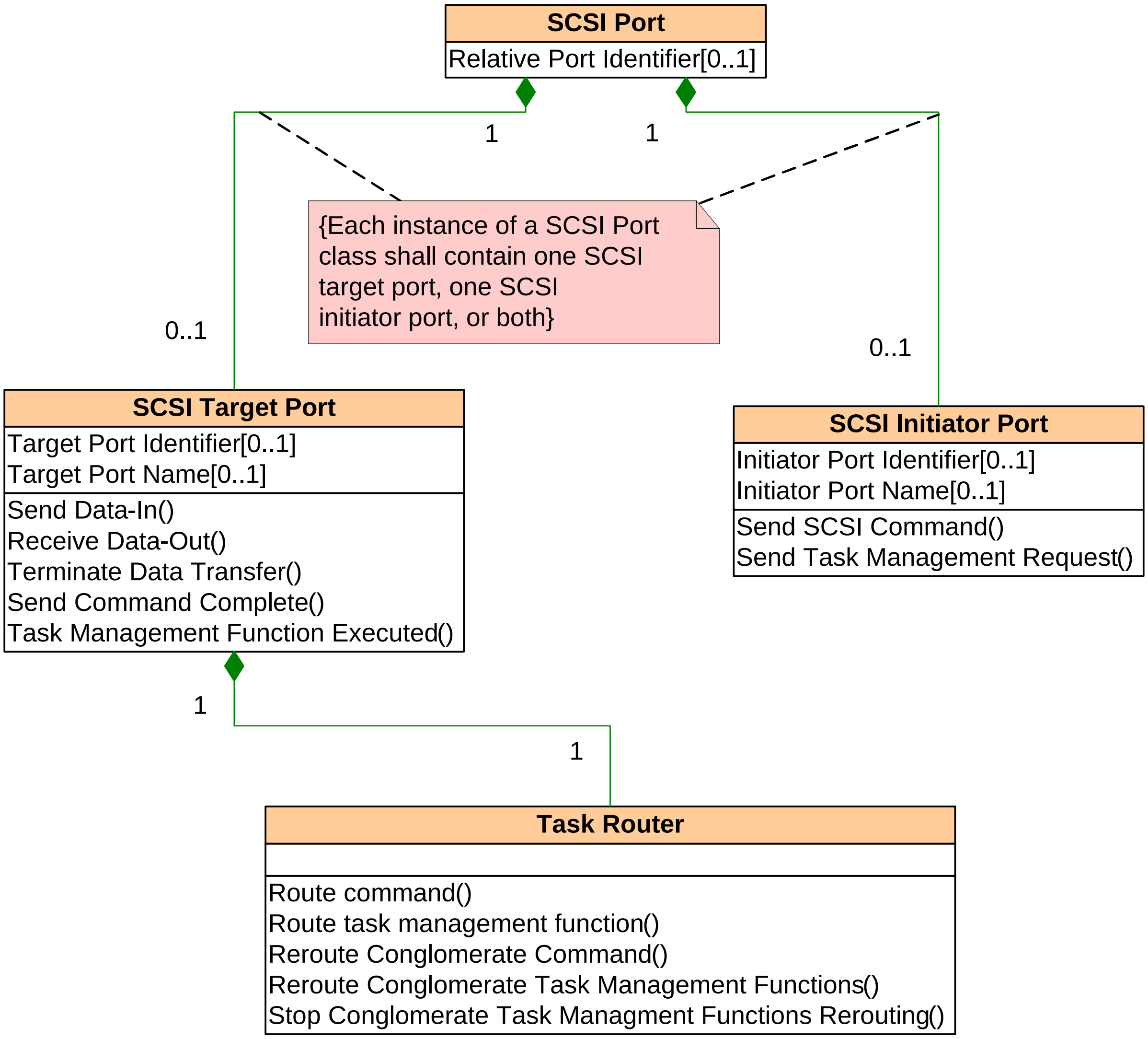


Figure 15 — SCSI Port class diagram

Each instance of a SCSI Port class shall contain:

a) one SCSI target port that shall contain:
   A) one task router;
b) one SCSI initiator port; or
c) both.

Instantiation requirements for the SCSI Target Port class are also described in 4.6.6.1.

4.6.5.2 Relative Port Identifier attribute

The Relative Port Identifier attribute identifies a SCSI target port or a SCSI initiator port relative to other SCSI ports in a SCSI target device and any SCSI initiator devices contained within that SCSI target device. A SCSI target device may assign relative port identifiers to its SCSI target ports and any SCSI initiator ports. If relative port identifiers are assigned, the SCSI target device shall assign each of its SCSI target ports and any SCSI initiator ports a unique relative port identifier from 1 to 65 535. SCSI target ports and SCSI initiator ports share the same number space.

The Device Identification VPD page (see SPC-4) and the SCSI Ports VPD page (see SPC-4) report relative port identifiers.

The relative port identifiers are not required to be contiguous. The relative port identifier for a SCSI port shall not change once assigned unless physical reconfiguration of the SCSI target device occurs.

4.6.6 SCSI Target Port class

4.6.6.1 SCSI Target Port class overview

The SCSI Target Port class (see figure 15) contains the:

a) Task Router class (see 4.6.7).

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

a) one task router.

Instantiation requirements for the SCSI Target Port class are also described in 4.6.6.1.

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

The SCSI Target Port class is associated with the SCSI Target Device class (see figure 16).

The SCSI target port invokes:

a) ~~a)~~ the Data-In Delivered operation (see 5.4.3.2.2) of a device server after sending data over the service delivery subsystem;
b) ~~b)~~ the Data-Out Received operation (see 5.4.3.3.2) of a device server after receiving data over the service delivery subsystem;
c) ~~c)~~ the Data Transfer Terminated operation (see 5.4.3.4.2) of a device server or a task manager after terminating data transfers;
d) ~~d)~~ the Route Command operation (see 4.6.7.2) of the task router in response to receiving a command over the service delivery subsystem; and
e) ~~e)~~ the Route Task Management Function operation (see 4.6.7.3) of the task router in response to receiving a task management function over the service delivery subsystem;

4.6.6.2 Target Port Identifier attribute

The Target Port Identifier attribute, if any, contains a target port identifier for a SCSI target port. The target port identifier is a value by which a SCSI target port is referenced within a SCSI domain.

If a transport protocol standard supports more than one SCSI target port within a SCSI domain, then that transport protocol standard shall define target port identifiers. If a transport protocol standard only supports one SCSI target port within a SCSI domain, then that transport protocol standard is not required to define target port identifiers.

The target port identifier may be reported in a target port name designation descriptor in the Device Identification VPD page (see SPC-4). If a SCSI target port has a target port identifier and a target port name see SPC-4 to determine which is reported.

4.6.6.3 Target Port Name attribute

A Target Port Name attribute, if any, contains an optional name of a SCSI target port that is world wide unique within the SCSI transport protocol of the SCSI domain of that SCSI target port. A SCSI target port shall have at most one name. A target port name shall never change and may be used for persistent identification of a SCSI target port.

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

The target port name may be reported in a target port name designation descriptor in the Device Identification VPD page (see SPC-4). If a SCSI target port has a target port identifier and a target port name see SPC-4 to determine which is reported.

4.6.6.4 Send Data-In operation

The Send Data-In operation implements the Send Data-In SCSI transport protocol service request (see 5.4.3.2.1) by sending data over the service delivery subsystem.

4.6.6.5 Receive Data-Out operation

The Receive Data-Out operation implements the Receive Data-Out SCSI transport protocol service request (see 5.4.3.3.1) by receiving data over the service delivery subsystem.

4.6.6.6 Terminate Data Transfer operation

The Terminate Data Transfer operation implements the Terminate Data Transfer SCSI transport protocol service request (see 5.4.3.4.1) by terminating data transfers.

4.6.6.7 Send Command Complete operation

The Send Command Complete operation implements the Send Command Complete SCSI transport protocol service response (see 5.4.2.4) by transmitting command complete information over the service delivery subsystem.

4.6.6.8 Task Management Function Executed operation

The Task Management Function Executed operation implements the Task Management Function Executed SCSI transport protocol service response (see 7.12.4) by transmitting task management function executed information over the service delivery subsystem.

4.6.7 Task Router class

4.6.7.1 Task Router class overview

The Task Router class (see figure 15) routes commands and task management functions.

In some transport protocols, the task router may check for overlapped command identifiers on commands (see 5.10).

4.6.7.2 Route Command operation

This operation is modeled by the following procedure call:

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

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the command (see 4.8).

CDB: Command descriptor block (see 5.2).

Task Attribute: A value specifying one of the task attributes defined in 8.6. For specific requirements on the Task Attribute argument see 5.1.

CRN: When a CRN argument is used, all commands on an I_T_L nexus include a CRN argument (see 5.1).

Command Priority: The priority assigned to the command (see 8.7).

First Burst Enabled: An argument specifying that a SCSI transport protocol specific number of bytes from the Data-Out Buffer are being delivered to the logical unit without waiting for the device server to invoke the Receive Data-Out SCSI transport protocol service.

The Route Command operation routes a command to a task manager as follows:

1) if the conditions specified in 4.6.7.4 are met, then a command associated with a valid subsidiary logical unit are rerouted to the administrative logical unit;
2) commands associated with~~addressed to~~ a valid logical unit are routed to the task manager in the specified logical unit by invoking the SCSI Command Received operation (see 5.4.2.3) of that task manager;
3) commands that are associated with an incorrect logical unit number in which the ADMINISTRATIVE ELEMENT field (see 4.7.6.2) specifies an administrative logical unit (see 4.6.11) are routed to the task manager in that administrative logical unit by invoking the SCSI Command Received operation of the task manager in the logical unit that is specified by setting the subsidiary element to zero (i.e., using the algorithm described in 4.6.10) without modifying the contents of the I_T_L_Q nexus input to the SCSI Command Received operation; and
4) commands associated with~~addressed to~~ an incorrect logical unit number are handled as described in 5.11.

4.6.7.3 Route Task Management Function operation

This operation is modeled by the following procedure call**:**

Route Task Management Function	**(IN (Nexus, Function Identifier))**

Input arguments:

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.8).

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

The Route Task Management Function operation routes a task management function to one or more task managers as follows:

1) if the conditions specified in 4.6.7.5 are met, then task management functions associated with a valid subsidiary logical unit may be rerouted to the administrative logical unit;
2) task management functions with I_T_L nexus scope (e.g., ABORT TASK SET, CLEAR TASK SET, CLEAR ACA, LOGICAL UNIT RESET, QUERY TASK SET, and QUERY ASYNCHRONOUS EVENT) or I_T_L_Q nexus scope (e.g., ABORT TASK and QUERY TASK) ~~addressed to~~associated with a valid logical unit are routed to the task manager in the specified logical unit by invoking the Task Management Request Received operation (see 7.12.3) of that task manager;
3) task management functions with an I_T nexus scope (e.g., I_T NEXUS RESET) are routed to the task manager in each logical unit by invoking the Task Management Request Received operation of each of those task managers; and
4) task management functions with I_T_L nexus scope or I_T_L_Q nexus scope associated with~~addressed to~~ an incorrect logical unit are handled as described in 7.12.~~ This operation is modeled by the following procedure call:~~

4.6.7.4 Reroute Conglomerate Command operation

This operation is modeled by the following procedure call:

Reroute Conglomerate Command	**(IN (LUN))**

Input arguments:

LUN: The logical unit number for the administrative logical unit in a logical unit conglomerate (see 4.6.10).

The processing of a Route Conglomerate Command operation causes the task router to reroute to the administrative logical unit (see 4.6.11) the next command received that is associated with any valid subsidiary logical unit (see 4.6.12) in the logical unit conglomerate whose administrative logical unit has specified the LUN.

The processing of a Reroute Conglomerate Command operation causes the Route Command operation (see 4.6.7.2) to reroute one of the received commands using the following method:

1) the I_T_L_Q nexus of every command processed by the Route Command operation is monitored for a command that is associated with one of the valid subsidiary logical units in the logical unit conglomerate whose administrative logical unit is specified by the LUN argument to the Reroute Conglomerate Command operation using the method described in 4.6.10 to detect the LUNs of valid subsidiary logical units;
2) the first command associated with a specified valid subsidiary logical unit is rerouted to the specified administrative logical unit by invoking the SCSI Command Received operation (see 4.6.20.2) of the administrative logical unit's task manager without modifying the contents of the I_T_L_Q nexus input to the Route Command operation; and
3) after a command is rerouted, no other commands are rerouted until an Reroute Conglomerate Command operation is processed.

The processing of a logical unit reset event (see 6.3.3) causes the task router to discard the task rerouting information stored for any Reroute Conglomerate Command operations that have not yet resulted in a command being rerouted.

Commands that are associated with an incorrect logical unit number (see 5.11) are not affected by the Reroute Conglomerate Command operation.

4.6.7.5 Reroute Conglomerate Task Management Functions operation

This operation is modeled by the following procedure call:

Reroute Conglomerate Task Management Functions **(IN (LUN))**

Input arguments:

LUN: The logical unit number for the administrative logical unit in a logical unit conglomerate (see 4.6.10).

Until stopped as described in this subclause, the processing of a Route Conglomerate Task Management Functions operation causes the task router to reroute to the administrative logical unit (see 4.6.11) all task management functions received that are associated with any valid subsidiary logical unit (see 4.6.12) in the logical unit conglomerate whose administrative logical unit has specified the LUN.

The processing of a Reroute Conglomerate Task Management Functions operation causes the Route Task Management Function operation (see 4.6.7.3) to reroute received task management functions using the following method:

1) the I_T_L_Q nexus of every task management function processed by the Route Task Management Function operation is monitored for a task management function that is associated with one of the valid subsidiary logical units in the logical unit conglomerate whose administrative logical unit is specified by the LUN argument to the Reroute Conglomerate Task Management Function operation using the method described in 4.6.10 to detect the LUNs of valid subsidiary logical units; and
2) any task management function associated with a specified valid subsidiary logical unit is rerouted to the specified administrative logical unit by invoking the Task Management Request Received operation (see 4.6.20.3) of administrative logical unit's task manager without modifying the contents of the I_T_L_Q nexus input to the Route Task Management Function operation.

The rerouting of task management functions is stopped by the processing of a:

a) Stop Conglomerate Task Management Functions Rerouting operation (see 4.6.7.6); or

b) logical unit reset event (see 6.3.3).

Task management functions that are associated with an incorrect logical unit number (see 5.11) are not affected by the Reroute Conglomerate Task Management Functions operation.

4.6.7.6 Stop Conglomerate Task Management Functions Rerouting operation

This operation is modeled by the following procedure call:

Stop Conglomerate Task Management Functions Rerouting **(IN (LUN))**

Input arguments:

LUN: The logical unit number for the administrative logical unit in a logical unit conglomerate (see 4.6.10).

The processing of a Stop Conglomerate Task Management Functions Rerouting operation ends the rerouting of task management functions:

a) that are associated with any valid subsidiary logical unit (see 4.6.12) in the logical unit conglomerate whose administrative logical unit (see 4.6.11) has specified the LUN; and
b) for which rerouting was begun by a prior Reroute Conglomerate Task Management Functions operation (see 4.6.7.5).

4.6.8 SCSI Initiator Port class

4.6.8.1 SCSI Initiator Port class overview

The SCSI Initiator Port class (see figure 15) connects SCSI initiator devices to a service delivery subsystem.

The SCSI Initiator Port class is associated with the SCSI Initiator Device class (see figure 13).

The SCSI initiator port invokes:

a) a) the Command Complete Received operation (see 5.4.2.5) of an application client after receiving command complete information over the service delivery subsystem; and
b) b) the Received Task Management Function Executed operation (see 7.12.5) of an application client after receiving task management function executed information over the service delivery subsystem.

4.6.8.2 Initiator Port Identifier attribute

The Initiator Port Identifier attribute, if any, contains an initiator port identifier for a SCSI initiator port. The initiator port identifier is a value by which a SCSI initiator port is referenced within a SCSI domain.

If a transport protocol standard supports more than one SCSI initiator port within a SCSI domain, then that transport protocol standard shall define initiator port identifiers. If a transport protocol standard only supports one SCSI initiator port within a SCSI domain, then that transport protocol standard is not required to define initiator port identifiers.

The initiator port identifier is reported by methods outside the scope of this standard.

4.6.8.3 Initiator Port Name attribute

A Initiator Port Name attribute, if any, contains an optional name of a SCSI initiator port that is world wide unique within the SCSI transport protocol of the SCSI domain of that SCSI initiator port. A SCSI initiator port shall have at most one name. An initiator port name shall never change and may be used for persistent identification of a SCSI initiator port.

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

The initiator port name is reported by methods outside the scope of this standard.

4.6.8.4 Send SCSI Command operation

The Send SCSI Command operation implements the Send SCSI Command SCSI transport protocol service request (see 5.4.2.2) by sending a command over the service delivery subsystem.

4.6.8.5 Send Task Management Request operation

The Send Task Management Request operation implements the Send Task Management Request SCSI transport protocol service request (see 7.12.2) by sending a task management function over the service delivery subsystem.

4.6.9 SCSI Target Device class

The SCSI Target Device class (see figure 16) is a SCSI Device class that contains the:

a) Level 1 Hierarchical Logical Unit class (see 4.6.13) and;
b) Logical Unit Conglomerate class (see 4.6.10).

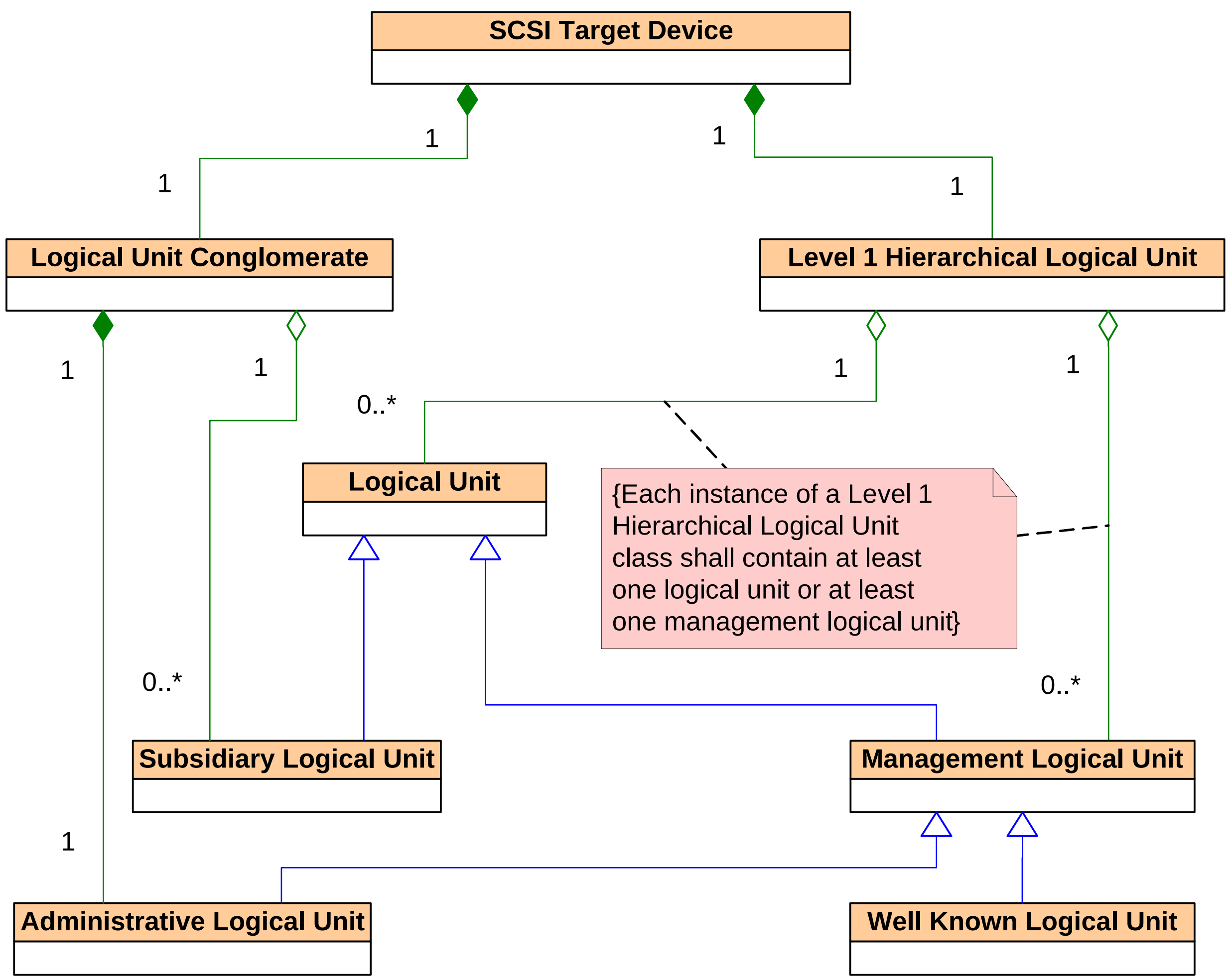


Figure 16 — SCSI Target Device class diagram

Each instance of the SCSI Target Device class shall contain the following objects:

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

   and

b) zero or more logical unit conglomerates that contain:
   A) one administrative logical unit; and
   B) zero or more subsidiary logical units.

The SCSI Target Device class is associated with the SCSI Target Port class (see figure 13).

4.6.10 Logical Unit Conglomerate class

The Logical Unit Conglomerate class (see figure 16) contains the:

a) Administrative Logical Unit class (see 4.6.11); and
b) Subsidiary Logical Unit class (see 4.6.12).

Each instance of the Logical Unit Conglomerate class shall contain:

a) one administrative logical unit; and
b) zero or more subsidiary logical units.

Management functionality (see 4.6.24) provided by the device server in the administrative logical unit may:

a) affect subsidiary logical units, if any, in the logical unit conglomerate when a management function is performed by the administrative logical unit's device server; and
b) represent the administrative logical unit and subsidiary logical units, if any, in the logical unit conglomerate when a condition is established that affects information returned to the application client (e.g., unit attention coalescing).

Management functionality that affects the logical unit conglomerate is requested from outside the SCSI target device (e.g., by an application client). Management functionality that represents the logical unit conglomerate may cause information (e.g., asymmetric logical unit access state) to be transferred outside the SCSI target device (e.g., to an application client).

The conditions that cause management functionality to be performed include:

a) requests (i.e., commands) sent by application clients; and
b) events and requests that are outside the scope of this standard (e.g., requests received from outside the SCSI domain).

The administrative logical unit and all subsidiary logical units, if any, in a logical unit conglomerate shall:

a) use the logical unit conglomerate LUN structure (see 4.7.6.2); and
b) set the LU_CONG bit to one in their standard INQUIRY data (see SPC-4).

The logical unit conglomerate LUN structure (see 4.7.6.2) is constructed to allow the determination of the LUN of the administrative logical unit in an logical unit conglomerate based on the LUN for a subsidiary logical unit in the same logical unit conglomerate. If the bits that specify the subsidiary logical unit in its LUN (e.g., the bits in the SUBSIDIARY ELEMENT field) are set to zero, then the resulting LUN is that of the administrative logical unit in the same logical unit conglomerate.

4.6.11 Administrative Logical Unit class

4.6.11.1 Administrative Logical Unit class overview

The Administrative Logical Unit class is a Management Logical Unit class (see 4.6.24) with the additional characteristics defined in 4.6.11.

The management functionality (see 4.6.24) provided by an administrative logical unit shall not affect logical units outside the logical unit conglomerate.

An administrative logical unit shall not:

a) contain the Dependent Logical Unit attribute (see 4.6.17.4); or
b) be a well-known logical unit (see 4.6.25).

4.6.11.2 Rerouted commands and task management functions

4.6.11.2.1 Overview

The task router (see 4.6.7) may route a command or task management function from subsidiary logical units to the administrative logical unit for the following reasons:

a) the command is associated with a subsidiary logical unit that is not valid (see 4.6.11.2.2);
b) the rerouting of a command was requested by the administrative logical unit (see 4.6.11.2.3); and
c) the rerouting of all task management functions was requested by the administrative logical unit (see 4.6.11.2.4).

The administrative logical unit's device server and task manager shall monitor the I_T_L_Q nexus argument for each received command and task management function to detect the delivery of a command or task management function that is associated with a subsidiary logical unit.

4.6.11.2.2 Commands rerouted due to incorrect logical unit selection

If task router processing causes an administrative logical unit's device server to receive a command associated with an incorrect logical unit number for a subsidiary logical unit (see 4.6.7.2), then the device server:

a) shall terminate the command with CHECK CONDITION status, with the sense key set to ILLEGAL REQUEST, and with the additional sense code set to SUBSIDIARY LOGICAL UNIT NOT CONFIGURED;
b) shall ignore the value of the NACA bit in the CDB; and
c) shall not establish an ACA condition in the administrative logical unit based on the delivery of a CHECK CONDITION related to an incorrect logical unit number for a subsidiary logical unit.

4.6.11.2.3 Command rerouting in response to an administrative logical unit request

A method that the administrative logical unit uses to manage its logical unit conglomerate (e.g., for unit attention coalescing (see 5.14.2)) is to request the task manager to reroute to the administrative logical unit's device server one command that an application client sends to a device server in one of the subsidiary logical units. The administrative logical unit's device server causes one command to be rerouted by invoking the Reroute Conglomerate Command operation (see 4.6.7.4) in the task router (see 4.6.7).

If a command associated with a subsidiary logical unit is rerouted as a result of the processing of a Reroute Conglomerate Command operation, then the administrative logical unit's device server processes the command as follows:

a) if:
    A) the command is allowed to terminate with CHECK CONDITION status;
    B) the subsidiary logical unit with which the command is associated is not in a condition that requires the return of a status that has a higher status precedence (see 5.3.3) than that of the highest precedence unit attention (see 5.14.1) in the administrative device server's coalesced unit attentions queue (see 5.14.2); and
    C) the subsidiary logical unit with which the command is associated does not have a unit attention condition in its queue that has a higher unit attention precedence (see 5.14.1) than that of the highest precedence unit attention in the administrative device server's coalesced unit attentions queue,

    then the administrative logical unit's device server shall:

    A) not begin processing for the command; and

B) terminate the command with CHECK CONDITION and set the sense data to indicate the appropriate logical unit conglomerate condition (e.g., a unit attention condition);

or

b) if:
   A) the command is not allowed to terminate with CHECK CONDITION status (e.g., an INQUIRY command);
   B) the subsidiary logical unit with which the command is associated is in a condition that requires the return of a status that has a higher status precedence (see 5.3.3) than that of the highest precedence unit attention condition (see 5.14.1) in the administrative device server's coalesced unit attentions queue (see 5.14.2); or
   C) the subsidiary logical unit with which the command is associated has a unit attention condition in its queue that has a higher unit attention precedence (see 5.14.1) than that of the highest precedence unit attention condition in the administrative device server's coalesced unit attentions queue,

   then the administrative logical unit's device server:

   1) shall invoke the Route Command operation (see 4.6.7.2) to route the command to the correct subsidiary logical unit; and
   2) may invoke the Reroute Conglomerate Command operation to request that the task router reroute another command associated with a subsidiary logical unit.

4.6.11.2.4 Task management function rerouting in response to an administrative logical unit request

A method that the administrative logical unit uses to manage its logical unit conglomerate (e.g., for unit attention coalescing (see 5.14.2)) is to request the task manager to reroute to the administrative logical unit's task manager zero or more task management function requests that an application client sends to a task manager in any of the subsidiary logical units. The administrative logical unit's device server causes zero or more task management function requests to be rerouted by invoking the following task router operations:

a) Reroute Conglomerate Task Management Functions operation (see 4.6.7.5) to begin the rerouting of task management function requests, if any; and
b) Stop Conglomerate Task Management Functions Rerouting operation (see 4.6.7.6) to stop the rerouting of task management function requests.

If a task management function request associated with a subsidiary logical unit is rerouted as a result of the processing of a Reroute Conglomerate Task Management Functions operation, then the administrative logical unit's task manager processes the task management function as follows:

a) if:
   A) the task management function is a QUERY ASYNCHRONOUS EVENT (see 7.10); and
   B) there is at least one unit attention condition in the administrative device server's coalesced unit attentions queue (see 5.14.2),

   then the administrative logical unit's task manager shall:

   A) return a service response of FUNCTION SUCCEEDED for the task management function;
   B) set the SENSE KEY field to UNIT ATTENTION in the Additional Response Information argument (see 7.10);
   C) set the ADDITIONAL SENSE CODE field and ADDITIONAL SENSE CODE QUALIFIER field in the Additional Response Information argument based on which one of the following unit attention conditions has the higher unit attention precedence (see 5.14.1):
      a) the highest precedence unit attention condition in administrative logical unit's coalesced unit attentions queue; or
      b) the highest precedence unit attention condition, if any, in the subsidiary logical unit's unit attention queue;

      and

   D) combine the following values to set the UADE DEPTH field in the Additional Response Information argument:
      a) the contents of the administrative logical unit's coalesced unit attentions queue;

   b) the contents of the unit attentions queue in the subsidiary logical unit with which the task management function is associated; and
   c) the presence of a deferred error condition in the subsidiary logical unit with which the task management function is associated;

or

b) if:
   A) the task management function is not a QUERY ASYNCHRONOUS EVENT (see 7.10); or
   B) there are no unit attention conditions in the administrative device server's coalesced unit attentions queue (see 5.14.2),

   then the administrative logical unit's task manager shall cause the task management function to be processed by invoking the Task Management Request Received operation (see 4.6.20.3) in the task manager of the subsidiary logical unit associated with the task management function.

4.6.12 Subsidiary Logical Unit class

The Subsidiary Logical Unit class is a Logical Unit class (see 4.6.17) that:

a) is contained in a logical unit conglomerate (see 4.6.10); and
b) is not an administrative logical unit (see 4.6.11).

A subsidiary logical unit shall not:

a) contain the Dependent Logical Unit attribute (see 4.6.17.4); or
b) be a well known logical unit (see 4.6.25).

4.6.13 Level 1 Hierarchical Logical Unit class

The Level 1 Hierarchical Logical Unit class (see figure 16 and figure 17) contains the:

a) Logical Unit class (see 4.6.17);
b) Management Logical Unit class (see 4.6.24);
c) Well Known Logical Unit class (see 4.6.25); and
d) Level 2 Hierarchical Logical Unit class (see 4.6.14).

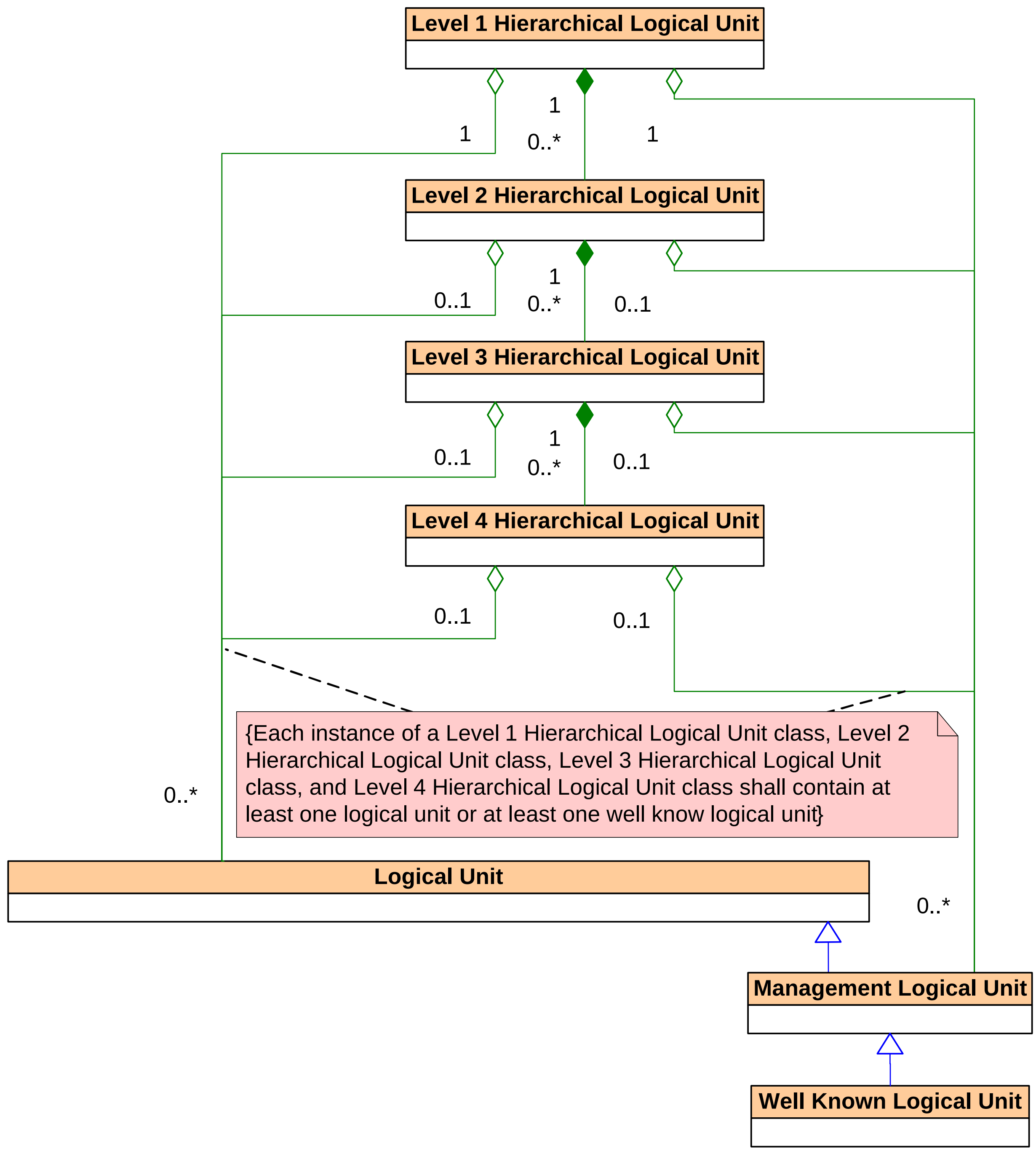


Figure 17 — Level 1 Hierarchical Logical Unit class

Each instance of a Level 1 Hierarchical Logical Unit class shall contain the following objects:

a) at least one logical unit or management logical unit;
b) zero or more logical units;
c) zero or more management logical units;
d) zero or more well known logical units;
e) zero or more level 2 hierarchical logical units;
f) zero or more level 3 hierarchical logical units; or
g) zero or more level 4 hierarchical logical units.

Logical units, management logical units, and well known logical units at each level in the hierarchical logical unit structure are referenced by one of the following address methods:

a) peripheral device address method (see 4.7.7.2);
b) flat space addressing method (see 4.7.7.3);
c) logical unit address method (see 4.7.7.4); or
d) extended logical unit addressing method (see 4.7.7.5).

All peripheral device addresses, except LUN 0 (see 4.7.4), default to vendor specific values. All addressable entities, except well known logical units (see 4.6.25), may default to vendor specific values or may be defined by an application client (e.g., by the use of SCC-2 configuration commands).

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

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

When using the peripheral device addressing method or the logical unit address method the SCSI domains accessed by these SCSI initiator ports are referred to as buses. A SCSI target device that has SCSI devices attached to these buses shall assign numbers, other than zero, to those buses. The bus numbers shall be used as components of the LUNs to the logical units attached to those buses, as described in 4.7.7.2 and 4.7.7.4.

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

4.6.14 Level 2 Hierarchical Logical Unit class

The Level 2 Hierarchical Logical Unit class (see figure 17) contains the:

a) Logical Unit class;
b) Management Logical Unit class;
c) Well Known Logical Unit class; and
d) Level 3 Hierarchical Logical Unit class (see 4.6.15).

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

All logical units, management logical units, and well known logical units contained within a level 2 hierarchical logical unit shall have a Dependent Logical Unit attribute (see 4.6.17.4).

4.6.15 Level 3 Hierarchical Logical Unit class

The Level 3 Hierarchical Logical Unit class (see figure 17) contains the:

a) Logical Unit class;
b) Management Logical Unit class;
c) Well Known Logical Unit class; and
d) Level 4 Hierarchical Logical Unit class (see 4.6.16).

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

All logical units, management logical units, and well known logical units contained within a level 3 hierarchical logical unit shall have a Dependent Logical Unit attribute (see 4.6.17.4).

4.6.16 Level 4 Hierarchical Logical Unit class

The Level 4 Hierarchical Logical Unit class (see figure 17) contains the:

a) Logical Unit class;
b) Management Logical Unit class; and
c) Well Known Logical Unit class.

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

All logical units, management logical units, and well known logical units contained within a level 4 hierarchical logical unit shall have a Dependent Logical Unit attribute (see 4.6.17.4).

4.6.17 Logical Unit class

4.6.17.1 Logical Unit class overview

The Logical Unit class (see figure 18) contains the:

a) Device Server class (see 4.6.18);
b) Task Manager class (see 4.6.20);
c) Copy Manager class (see 4.6.19);
d) Task Management Function class (see 4.6.23); and
e) Task Set class (see 4.6.21).

The Logical Unit class (see figure 18) may be substituted with the:

a) Management Logical Unit class (see 4.6.24);~~ or~~
b) Well Known Logical Unit class (see 4.6.17.1); or
c) Subsidiary Logical Unit (see 4.6.12).

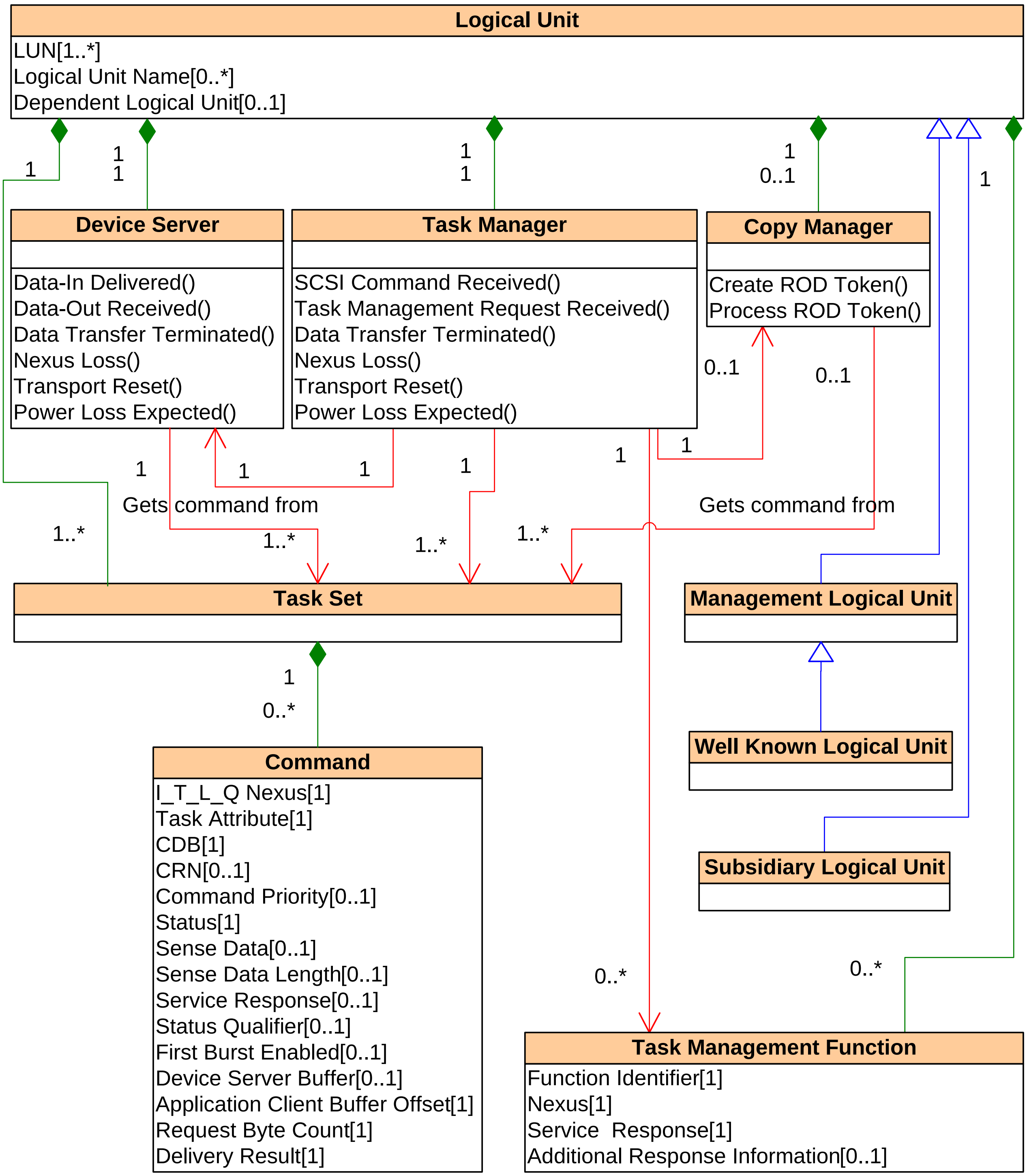


Figure 18 — Logical Unit class diagram

Each instance of a Logical Unit class shall contain the following objects:

a) one device server;
b) one task manager;
c) zero or one copy manager;
d) zero or more task management functions; and
e) one or more task sets each of which that shall contain:
   A) zero or more commands.

Instantiation requirements for the Task Set class are also described in 4.6.21.

The logical unit is the class to which commands are sent. One of the logical units within the SCSI target device shall be accessed using the LUN zero or the REPORT LUNS well known logical unit number.

If the logical unit inventory changes for any reason (e.g., completion of initialization, removal of a logical unit, or creation of a logical unit), then the device server shall establish a unit attention condition (see 5.14) for the SCSI initiator port associated with every I_T nexus, with the additional sense code set to REPORTED LUNS DATA HAS CHANGED.

Data contained within a logical unit may be duplicated in whole or part on a different logical unit. The synchronization of that data between multiple logical units is outside the scope of this standard.

4.6.17.2 LUN attribute

The LUN attribute identifies the logical unit within a SCSI target device when accessed by a SCSI target port. If any logical unit within the scope of a SCSI target device includes one or more dependent logical units (see 4.6.17.4) in its composition, then all LUNs within the scope of the SCSI target device shall have the format described in 4.7.6.3. If there are no dependent logical units within the scope of the SCSI target device, the LUNs should have the format described in 4.7.5.

The 64-bit or 16-bit quantity called a LUN is the LUN attribute defined by this standard. The fields containing the acronym LUN that compose the LUN attribute are historical nomenclature anomalies, not LUN attributes. LUN attributes having different values represent different logical units, regardless of any implications to the contrary in 4.7 (e.g., LUN 00000000 00000000h is a different logical unit from LUN 40000000 00000000h and LUN 00FF0000 00000000h is a different logical unit from LUN 40FF0000 00000000h).

LUN(s) are required as follows:

a) if access controls (see SPC-4) are not in effect, one LUN per logical unit; or
b) if access controls are in effect, one LUN per SCSI initiator port that has access rights plus one default LUN per logical unit.

See 4.7 for a definition of:

a) the construction of LUNs to be used by SCSI target devices; and
b) incorrect logical unit numbers.

4.6.17.3 Logical Unit Name attribute

The Logical Unit Name attribute identifies a name (see 3.1.69) for a logical unit that is not a well known logical unit. A logical unit name shall be world wide unique. A logical unit name shall never change and may be used for persistent identification of a logical unit.

Logical unit name(s) are required as follows:

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

The name used to identify the logical unit is the logical unit name designation descriptor in the Device Identification VPD page (see SPC-4).

4.6.17.4 Dependent Logical Unit attribute

The Dependent Logical Unit attribute identifies a logical unit that is:

a) addressed via a hierarchical logical unit that resides at a lower numbered level in the hierarchy (i.e., when more than one level of hierarchy is used, no logical unit within level 1 contains a Dependent Logical Unit attribute while all logical units within level 2, level 3, and level 4 do contain a Dependent Logical Unit attribute); or
b) addressed with the logical unit addressing method (see 4.7.7.4) via another logical unit at the same numbered level in the hierarchy.

Any instance of a Logical Unit class that contains a Dependent Logical Unit attribute shall utilize the hierarchical LUN structure defined in 4.7.6.3. If any logical unit within a SCSI target device includes a Dependent Logical Unit attribute:

a) all logical units within the SCSI target device shall format all LUNs as described in 4.7.6.3; and
b) LUN zero or the REPORT LUNS well known logical unit (see SPC-4) shall set the HISUP bit to one in the standard INQUIRY data.

4.6.18 Device Server class

4.6.18.1 Device Server class overview

a) The Device Server class (see figure 18) processes commands (see 4.6.22) from the task set (see 4.6.21).

The device server invokes:

a) the Send Data-In operation (see 5.4.3.2.1) of a SCSI target port to send data to the data-in buffer;
b) the Receive Data-Out operation (see 5.4.3.3.2) of a SCSI target port to receive data from the data-out buffer;
c) the Terminate Data Transfer operation (see 5.4.3.4.1) of a SCSI target port to terminate data transfers for a command that it is aborting; and
d) the Send Command Complete operation (see 5.4.2.4) of a SCSI target port to complete processing a command.

4.6.18.2 Data-In Delivered operation

The Data-In Delivered operation implements the Data-In Delivered SCSI transport protocol service confirmation (see 5.4.3.2.2) by completing processing of sent data.

4.6.18.3 Data-Out Received operation

The Data-Out Received operation implements the Data-Out Received SCSI transport protocol service confirmation (see 5.4.3.3.1) by completing processing of received data.

4.6.18.4 Data Transfer Terminated operation

The Data Transfer Terminated operation implements the Data Transfer Terminated SCSI transport protocol service confirmation (see 5.4.3.4.1) by completing processing of a data transfer termination.

4.6.18.5 Nexus Loss operation

The Nexus Loss operation implements the Nexus Loss SCSI transport protocol service indication (see 6.4.2) by processing an I_T nexus loss (see 6.3.4).

4.6.18.6 Transport Reset operation

The Transport Reset operation implements the Transport Reset SCSI transport protocol service indication (see 6.4.3) by processing a hard reset (see 6.3.2).

4.6.18.7 Power Loss Expected operation

The Power Loss Expected operation implements the Power Loss Expected SCSI transport protocol service indication (see 6.4.4) by processing a power loss expected (see 6.3.5).

4.6.19 Copy Manager class

4.6.19.1 Copy Manager class overview

The Copy Manager class (see figure 18) is a kind of application client and a kind of device server (see figure 11) that processes third-party copy commands (see SPC-4) from the task set (see 4.6.21) and their associated copy operations (see SPC-4).

4.6.19.2 Create ROD Token operation

The Create ROD Token operation creates a ROD token (see SPC-4) and returns it to an application client as a representation of specified data that may be exchanged with other application clients.

4.6.19.3 Process ROD Token operation

The Process ROD Token operation converts a ROD token (see SPC-4) received from an application client into the data represented by the ROD token.

4.6.20 Task Manager class

4.6.20.1 Task Manager class overview

a) The Task Manager class (see figure 18) processes task management functions (see 4.6.23).

The task manager invokes:

a) the Terminate Data Transfer operation (see 5.4.3.4.1) of a SCSI target port to terminate data transfers for a command that the task manager is aborting; and
b) the Task Management Function Executed operation (see 7.12.4) of a SCSI target port to complete processing a task management function.

4.6.20.2 SCSI Command Received operation

The SCSI Command Received operation implements the SCSI Command Received SCSI transport protocol service indication (see 5.4.2.3).

To process the SCSI Command Received operation, the task manager creates a command (see 4.6.20) based on the Nexus, CDB, Task Attribute, CRN, Command Priority, and First Burst Enabled arguments and places the command into a task set (see 4.6.21).

4.6.20.3 Task Management Request Received operation

The Task Management Request Received operation implements the Task Management Request Received SCSI transport protocol service indication (see 7.12.3).

To process the Task Management Request Received operation, the task manager creates a task management function (see 4.6.23) based on the Nexus and Function Identifier arguments.

4.6.20.4 Data Transfer Terminated operation

The Data Transfer Terminated operation implements the Data Transfer Terminated SCSI transport protocol service confirmation (see 5.4.3.4.2) by completing processing of a data transfer termination.

4.6.20.5 Nexus Loss operation

The Nexus Loss operation implements the Nexus Loss SCSI transport protocol service indication (see 6.4.2) by processing an I_T nexus loss (see 6.3.4).

4.6.20.6 Transport Reset operation

The Transport Reset operation implements the Transport Reset SCSI transport protocol service indication (see 6.4.3) by processing a hard reset (see 6.3.2).

4.6.20.7 Power Loss Expected operation

The Power Loss Expected operation implements the Power Loss Expected SCSI transport protocol service indication (see 6.4.4) by processing a power loss expected (see 6.3.5).

4.6.21 Task Set class

The Task Set class (see figure 18) contains the:

a) Command class (see 4.6.22).

Each instance of a Task Set class shall contain the following objects:

a) zero or more commands.

Instantiation requirements for the Task Set class are also described in 4.6.17.1.

The interactions among the commands in a task set are determined by the requirements for task set management specified in clause 8 and the ACA requirements specified in 5.8. 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-4).

4.6.22 Command class

4.6.22.1 Command class overview

The Command class (see figure 18) represents a request to a logical unit to do work.

The command persists until deleted. For an example of the processing for a command see 5.7.

4.6.22.2 I_T_L_Q Nexus attribute

The I_T_L_Q Nexus attribute contains the I_T_L_Q nexus of the command (see 4.8).

4.6.22.3 Task Attribute attribute

A Task Attribute attribute (see 8.6) contains the task attribute (e.g., SIMPLE task attribute, ORDERED task attribute, HEAD OF QUEUE task attribute, ACA task attribute) of a command.

4.6.22.4 CDB attribute

The CDB attribute contains a CDB (see 5.2 and SPC-4) that defines the work to be performed by a logical unit.

4.6.22.5 CRN attribute

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

4.6.22.6 Command Priority attribute

The Command Priority attribute, if any, contains the priority of the command (see 8.7).

4.6.22.7 Status attribute

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

4.6.22.8 Sense Data attribute

The Sense Data attribute, if any, contains the sense data for the completed command (see 5.4.2.5).

4.6.22.9 Sense Data Length attribute

The Sense Data Length attribute, if any, contains the length of the sense data for the completed command (see 5.4.2.5).

4.6.22.10 Service Response attribute

The Service Response attribute, if any, contains the service response for the completed command (see 5.4.2.5).

4.6.22.11 Status Qualifier attribute

The Status Qualifier attribute, if any, contains additional status information for the completed command (see 5.3.2 and 5.4.2.5).

4.6.22.12 First Burst Enabled attribute

The First Burst Enabled attribute, if any, specifies that first burst for the command is enabled (see 5.4.2.5).

4.6.22.13 Device Server Buffer attribute

The Device Server Buffer attribute, if any, contains the Device Server Buffer argument used for Send Data-In procedure calls (see 5.4.3.2.1) and Receive Data-Out procedure calls (see 5.4.3.3.1).

4.6.22.14 Application Client Buffer Offset attribute

The Application Client Buffer Offset attribute contains the Application Client Buffer Offset argument used for Send Data-In procedure calls (see 5.4.3.2.1) and Receive Data-Out procedure calls (see 5.4.3.3.1).

4.6.22.15 Request Byte Count attribute

The Request Byte Count attribute contains the Request Byte Count argument used for Send Data-In procedure calls (see 5.4.3.2.1) and Receive Data-Out procedure calls (see 5.4.3.3.1).

4.6.22.16 Delivery Result attribute

The Delivery Result attribute contains the Delivery Result argument from a Data-In Delivered procedure call (see 5.4.3.2.1) or a Data-Out Received procedure call (see 5.4.3.3.1).

4.6.23 Task Management Function class

4.6.23.1 Task Management Function class overview

The Task Management Function class (see figure 18) represents a SCSI task management function (see clause 7).

The task management function persists until deleted. For an example of the processing for a task management function see 7.13.

4.6.23.2 Function Identifier attribute

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

4.6.23.3 Nexus attribute

The Nexus attribute identifies the nexus affected by the task management function (see 4.8).

4.6.23.4 Service Response attribute

The Service Response attribute contains the service response (see 7.12.4).

4.6.23.5 Additional Response Information attribute

The Additional Response Information attribute, if any, contains any additional response information for the task management function (see clause 7).

4.6.24 Management Logical Unit class

The Management Logical Unit class (see figure 18) is a Logical Unit class (see 4.6.17.1) with the additional characteristics defined in this subclause.

A management logical unit:

a) shall support access to management functionality; and

b) may support other features only for the purposes of device management.

4.6.25 Well Known Logical Unit class

The Well Known Logical Unit class (see figure 18) is a Management Logical Unit class (see 4.6.24) with the additional characteristics defined in this subclause.

Well known logical units are addressed using the well known logical unit addressing method (see 4.7.7.5.1) of extended logical unit addressing (see 4.7.7.5). Each well known logical unit has a well known logical unit number (W-LUN). W-LUN values are defined in SPC-4.

If a SCSI target port receives a command or a task management function specifying a W-LUN and the well known logical unit specified by the W-LUN does not exist, the task router shall follow the rules for processing an incorrect logical unit number described in 5.11 and 7.12.

A well known logical unit shall support all the commands defined for it.

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

All well known logical units:

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

NOTE 3 - A SCSI target device may have multiple SCSI device names if the SCSI target device supports multiple SCSI transport protocols (see 4.6.9).

The name used to identify the well known logical unit is indicated in the Device Identification VPD pages designation descriptor for SCSI target devices (see SPC-4).

4.6.26 SCSI Initiator Device class

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

a) Application Client class (see 4.6.27).

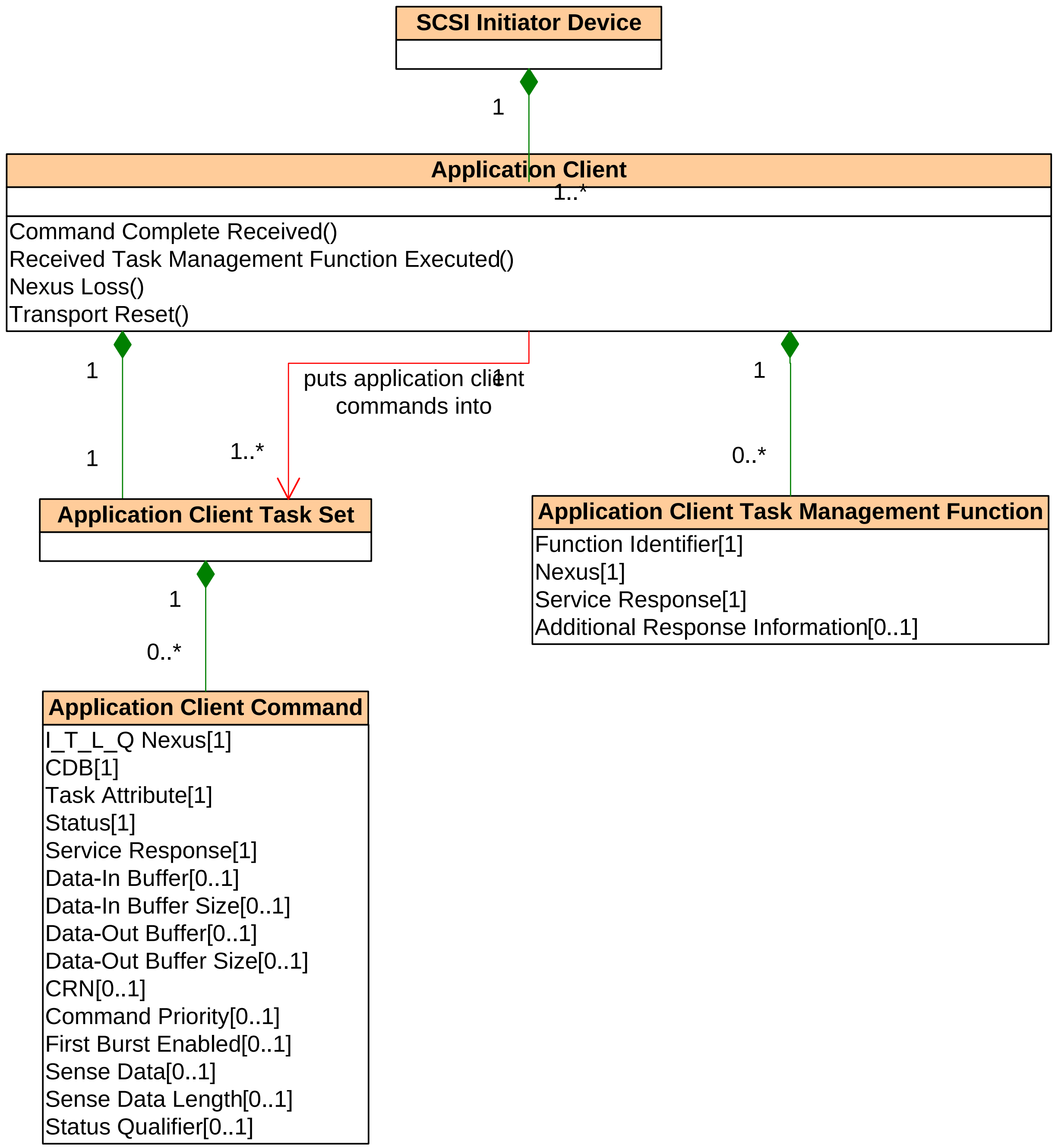


Figure 19 — SCSI Initiator Device class diagram

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

a) one or more application clients that contain:
   A) zero or more application client task management functions; and
   B) one application client task set.that contains:.
      a) zero or more application client commands.

Instantiation requirements for the Application Client class are also described in 4.6.27. Instantiation requirements for the Application Client Task Set class are also described in 4.6.29

The SCSI Initiator Device class is associated with the SCSI Initiator Port class (see figure 19).

4.6.27 Application Client class

4.6.27.1 Application Client class overview

An Application Client class (see figure 19) contains the:

a) Application Client Task Management Function class (see 4.6.28); and
b) Application Client Task Set class (see 4.6.29).

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

a) one application client task set; and
b) zero or more application client task management functions.

Instantiation requirements for the Application Client class are also described in 4.6.27.

An application client originates a command by:

a) creating an application client command (see 4.6.30) and placing that application client command into the application client task set; and
b) invoking the Send SCSI Command operation (see 5.4.2.2) of a SCSI initiator port.

The application client deletes an application client command from the application client task set after determining that the command has completed according to the command lifetime rules in 5.5 (e.g., processing the Command Complete Received operation).

An application client originates a task management function by:

a) creating an application client task management function (see 4.6.28); and
b) invoking the Send Task Management Request operation (see 7.12.2) of a SCSI initiator port.

The application client deletes an application client task management function after determining that the task management function has completed according to the task management function lifetime rules in 7.11 (e.g., processing the Received Task Management Function Executed operation).

The application client may request processing of a task management function for:

a) a logical unit through a request directed to the task manager within the logical unit; or
b) all logical units known by a task router through a request directed to the task router within the target port.

The interactions between the task manager, task router, and the application client when a task management request is processed are shown in 7.13.

4.6.27.2 Command Complete Received operation

The Command Complete Received operation implements the Command Complete Received SCSI transport protocol service confirmation (see 5.4.2.5) by completing the processing of a command.

4.6.27.3 Received Task Management Function Executed operation

The Received Task Management Function Executed operation implements the Received Task Management Function Executed SCSI transport protocol service request (see 7.12.5) by completing the processing of a task management function.

4.6.27.4 Nexus Loss operation

The Nexus Loss operation implements the Nexus Loss SCSI transport protocol service indication (see 6.4.2) by processing an I_T nexus loss (see 6.3.4).

4.6.27.5 Transport Reset operation

The Transport Reset operation implements the Transport Reset SCSI transport protocol service indication (see 6.4.3) by processing a hard reset (see 6.3.2).

4.6.28 Application Client Task Management Function class

4.6.28.1 Application Client Task Management Function class overview

The Application Client Task Management Function class (see figure 19) represents a SCSI task management function (see clause 7).

4.6.28.2 Function Identifier attribute

The Function Identifier attribute contains a function identifier (see 7.12).

4.6.28.3 Nexus attribute

The Nexus attribute contains the nexus affected by the task management function (see 4.8).

4.6.28.4 Service Response attribute

The Service Response attribute contains the service response (see clause 7).

4.6.28.5 Additional Response Information attribute

The Additional Response Information attribute, if any, contains any additional response information for the task management function (see clause 7).

4.6.29 Application Client Task Set class

The Application Client Task Set class (see figure 19) contains the:

a) Application Client Command class (see 4.6.30).

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

a) zero or more application client commands.

Instantiation requirements for the Application Client Task Set class are also described in 4.6.29.

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

4.6.30 Application Client Command class

4.6.30.1 Application Client Command class overview

The Application Client Command class (see figure 19) represents a request to a logical unit to do work (see clause 5). A new command causes the creation of an application client command. The application client command persists until a command complete response is received or until the command is completed by a task management function or exception condition. For an example of the processing for a command see 5.7.

4.6.30.2 I_T_L_Q Nexus attribute

The I_T_L_Q Nexus attribute contains the I_T_L_Q nexus (see 4.8) of the command (see 5.4.2.2).

4.6.30.3 CDB attribute

The CDB attribute contains a CDB (see 5.2 and SPC-4) that defines the work to be performed by a logical unit (see 5.4.2.2).

4.6.30.4 Task Attribute attribute

The Task Attribute attribute (see 8.6) contains the task attribute (e.g., SIMPLE task attribute, ORDERED task attribute, HEAD OF QUEUE task attribute, ACA task attribute) of a command (see 5.4.2.2).

4.6.30.5 Status attribute

The Status attribute contains the status (see 5.3) of the completed command (see 5.4.2.5)

4.6.30.6 Service Response attribute

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

4.6.30.7 Data-In Buffer attribute

The Data-In Buffer attribute, if any, contains the Data-In Buffer argument from an Execute Command procedure call (see 5.4.2.5).

4.6.30.8 Data-In Buffer Size attribute

The Data-In Buffer Size attribute, if any, contains the Data-In Buffer Size argument from an Execute Command procedure call (see 5.4.2.2).

4.6.30.9 Data-Out Buffer attribute

The Data-Out Buffer attribute, if any, contains the Data-Out Buffer argument from an Execute Command procedure call (see 5.4.2.2).

4.6.30.10 Data-Out Buffer size attribute

The Data-Out Buffer Size attribute, if any, contains the Data-Out Buffer Size argument from an Execute Command procedure call (see 5.4.2.2).

4.6.30.11 CRN attribute

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

4.6.30.12 Command Priority attribute

The Command Priority attribute, if any, contains the priority (see 8.7) of the command (see 5.4.2.2).

4.6.30.13 First Burst Enabled attribute

The First Burst Enabled attribute, if any, specifies that first burst for the command is enabled (see 5.4.2.2).

4.6.30.14 Sense Data attribute

The Sense Data attribute, if any, contains the sense data (see 5.13) for the completed command (see 5.4.2.5).

4.6.30.15 Sense Data Length attribute

The Sense Data Length attribute, if any, contains the length of the sense data (see 5.13) for the completed command (see 5.4.2.5).

4.6.30.16 Status Qualifier attribute

The Status Qualifier attribute, if any, contains additional status information for the completed command (see 5.3.2 and 5.4.2.5).

4.7 Logical unit number (LUN)

4.7.1 Introduction

Subclause 4.7 defines the construction of LUNs to be used by SCSI target devices. Application clients should use only those LUNs returned by a REPORT LUNS command (see SPC-4). The task router shall respond to incorrect logical unit numbers (i.e., LUNs other than those returned by a REPORT LUNS command with the SELECT REPORTS field set to 02h) as specified in 5.11 and 7.12.

4.7.2 Logical unit representation format

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

When displaying a single level LUN structure using the peripheral device addressing method (see table 11) or a single level LUN structure using the flat space addressing method (see table 12), an application client may display the value as a single 2-byte value representing only the first level LUN (e.g., 40 01h). A separator (e.g., space, dash, or colon) may be included between each byte.

When displaying a single level LUN structure using the extended flat space addressing method (see table 13), an application client may display the value as a single 4-byte value representing only the first level LUN (e.g., D2 00 00 01h). A separator (e.g., space, dash, or colon) may be included between each byte, or between each two bytes (e.g., D200 0001h).

When displaying a single level LUN structure using the long extended flat space addressing method (see table 14), an application client may display the value as a single 6-byte value representing only the first level LUN (e.g., E2 00 00 01 00 01h). A separator (e.g. space, dash, or colon) may be included between each byte, or between each two bytes (e.g., E200 0001 0001h).

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

4.7.3 LUNs overview

All LUN 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-4) when any LUN format described in this standard is used. Non-hierarchical formats are outside the scope of this standard.

The LUN identifier defined by a transport protocol standard shall contain 64 bits or 16 bits. For transport protocol standards that define 16-bit LUN identifiers:

a) the two bytes shall use an addressing method (see 4.7.6.3) with a length of two bytes (e.g., see table 24, table 25, table 26, or table 29); and
b) if the peripheral device addressing method (see 4.7.7.2) is used, then the BUS IDENTIFIER field shall be set to 00h.

All LUN identifiers defined by command standards shall contain 64 bits. When a 16-bit transport protocol LUN identifier is contained in a 64-bit LUN identifier defined by a command standard:

a) the 16 bits of the LUN identifier defined by the transport protocol standard shall be placed in the high order 16 bits of the 64-bit LUN identifier; and
b) all other bits of the 64-bit LUN identifier shall be set to zero.

4.7.4 Minimum LUN addressing requirements

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

The responses to commands sent to unsupported logical units are defined in 5.11. The response to task management functions sent to unsupported logical units is defined in 7.1.

4.7.5 Single level LUN structure

Table 11 describes a single level subset of the format described in 4.7.6.3 for SCSI target devices that contain 256 or fewer logical units.

Table 11 — Single level LUN structure using peripheral device addressing method

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

Byte 2 through byte 7 in an 8-byte single level LUN structure using the peripheral device addressing method shall each contain 00h (see table 11). The value in the TARGET OR LUN field shall address a single level logical unit and be between 0 and 255, inclusive. A value of 00b in the ADDRESS METHOD field specifies peripheral device addressing (see 4.7.6.3). A value of 00h in the BUS IDENTIFIER field specifies the current level (see 4.7.7.2).

Table 12 describes a single level subset of the format described in 4.7.6.3 for SCSI target devices that contain 16 384 or fewer logical units.

Table 12 — Single level LUN structure using flat space addressing method

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

Byte 2 through byte 7 in an 8-byte single level LUN structure using the flat space addressing method shall each contain 00h (see table 12). The value in the FLAT SPACE LUN field shall be between 0 and 16 383, inclusive. A value of 01b in the ADDRESS METHOD field specifies flat space addressing (see 4.7.7.3) at the current level.

Table 13 describes a single level subset of the format described in 4.7.6.3 for SCSI target devices that contain more than 16 384 logical units.

Table 13 — Single level LUN structure using extended flat space addressing method

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

Byte 4 through byte 7 in an 8-byte single level LUN structure using the extended flat space addressing method shall each contain 00h (see table 13). The value in the EXTENDED FLAT SPACE LUN field shall be between 0 and 16 777 215, inclusive. A value of 11b in the ADDRESS METHOD field with a value of 2h in the EXTENDED ADDRESS METHOD field specifies extended flat space addressing (see 4.7.7.5.2) at the current level. A value of 01b in the LENGTH field specifies that the LUN specified in the EXTENDED FLAT SPACE LUN field is three bytes in length.

Table 14 describes a single level subset of the format described in 4.7.6.3 for SCSI target devices that contain more than 16 777 216 logical units.

Table 14 — Single level LUN structure using long extended flat space addressing method

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

Byte 6 and byte 7 in an 8-byte single level LUN structure using the long extended flat space addressing method shall each contain 00h (see table 13.1). The value in the LONG EXTENDED FLAT SPACE LUN field shall be between 0 and 1 099 511 627 775, inclusive. A value of 11b in the ADDRESS METHOD field with a value of 2h in the EXTENDED ADDRESS METHOD field specifies extended flat space addressing (see 4.7.7.5.2) at the current level. A value of 10b in the LENGTH field specifies that the LUN specified in the LONG EXTENDED FLAT SPACE LUN field is five bytes in length.

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

If a SCSI target device contains 256 or fewer logical units, none of which are dependent logical units (see 4.6.17.4), then the SCSI target device's LUNs:

a) should have the format shown in table 11 (i.e., peripheral device addressing);
b) may have the format shown in table 12 (i.e., flat space addressing);
c) may have the format shown in table 13 (i.e., extended flat space addressing); or
d) may have the format shown in table 14 (i.e., long extended flat space addressing).

If a SCSI target device contains more than 256 logical units and 16 384 or fewer logical units, none of which are dependent logical units (see 4.6.17.4), then the SCSI target device's LUNs:

a) should have the format shown in table 12 (i.e., flat space addressing);
b) may have the format shown in table 13 (i.e., extended flat space addressing);
c) may have the format shown in table 14 (i.e., long extended flat space addressing); or
d) may have the format shown in table 11 (i.e., peripheral device addressing) for up to 256 of the logical units within the SCSI target device.

If a SCSI target device contains more than 16 384 logical units, none of which are dependent logical units (see 4.6.17.4), then the SCSI target device's LUNs:

a) should have the format shown in table 13 (i.e., extended flat space addressing);
b) may have the format shown in table 14 (i.e., long extended flat space addressing);
c) may have the format shown in table 12 (i.e., flat space addressing) for up to 16 384 of the logical units within the SCSI target device; or
d) may have the format shown in table 11 (i.e., peripheral device addressing) for up to 256 of the logical units within the SCSI target device.

If a SCSI target device contains more than 16 777 216 logical units, none of which are dependent logical units (see 4.6.17.4), then the SCSI target device's LUNs:

a) should have the format shown in table 14 (i.e., long extended flat space addressing);
b) may have the format shown in table 13 (i.e., extended flat space addressing) for up to 16 777 216 of the logical units within the SCSI target device;
c) may have the format shown in table 12 (i.e., flat space addressing) for up to 16 384 of the logical units within the SCSI target device; or
d) may have the format shown in table 11 (i.e., peripheral device addressing) for up to 256 of the logical units within the SCSI target device.

4.7.6 Complex LUN structures

4.7.6.1 Introduction to complex LUN structures

The eight byte LUN structures that represent complex configurations of logical units (e.g., logical units in a multilevel hierarchy) are summarized in table 15.

Table 15 — Complex LUN structures

LUN structure	Associated class	Reference
Conglomerate	Logical Unit Conglomerate class (see 4.6.10)	4.7.6.2
Hierarchical	Level 1 Hierarchical Logical Unit class (see 4.6.13)	4.7.6.3

4.7.6.2 Logical unit conglomerate LUN structure

The logical unit conglomerate LUN structure is shown in table 16. Each logical unit conglomerate LUN structure shall contain eight bytes.

Table 16 — Logical unit conglomerate LUN structure

Bit / Byte	7	6	5	4	3	2	1	0
0	ADMINISTRATIVE ELEMENT (see table 17)							
•••								
i								
i+1	SUBSIDIARY ELEMENT (see table 17)							
•••								
k								
k+1	PAD (if needed)							
•••								
7								

The ADMINISTRATIVE ELEMENT field contains addressing information for an administrative logical unit (see 4.6.11). The ADMINISTRATIVE ELEMENT field has a length of two, four, or six bytes. The format of the ADMINISTRATIVE ELEMENT field is shown in table 17.

The SUBSIDIARY ELEMENT field contains:

a) zero, when the logical unit conglomerate LUN structure identifies the administrative logical unit specified by the ADMINISTRATIVE ELEMENT field; or
b) non-zero logical unit addressing information for a subsidiary logical unit (see 4.6.12) in the same logical unit conglomerate as the administrative logical unit specified by the ADMINISTRATIVE ELEMENT field.

The SUBSIDIARY ELEMENT field has a length of two, four, or six bytes. The format of the SUBSIDIARY ELEMENT field is shown in table 17.

If a SUBSIDIARY ELEMENT field does not contain logical unit addressing information for a subsidiary logical unit, then that field shall be set to zero.

The PAD field shall contain zero to four bytes set to zero such that the total length of the logical unit conglomerate LUN structure is eight bytes.

The format of addressing fields in the logical unit conglomerate LUN structure is shown in table 17.

Table 17 — Format of addressing fields in the logical unit conglomerate LUN structure

Bit / Byte	7	6	5	4	3	2	1	0
k	ADDRESS METHOD		ADDRESS METHOD SPECIFIC					
k+m								

The ADDRESS METHOD field defines the contents of the ADDRESS METHOD SPECIFIC field. See table 18 for the address methods defined for the ADDRESS METHOD field in the logical unit conglomerate LUN structure.

Table 18 — ADDRESS METHOD field in the logical unit conglomerate LUN structure

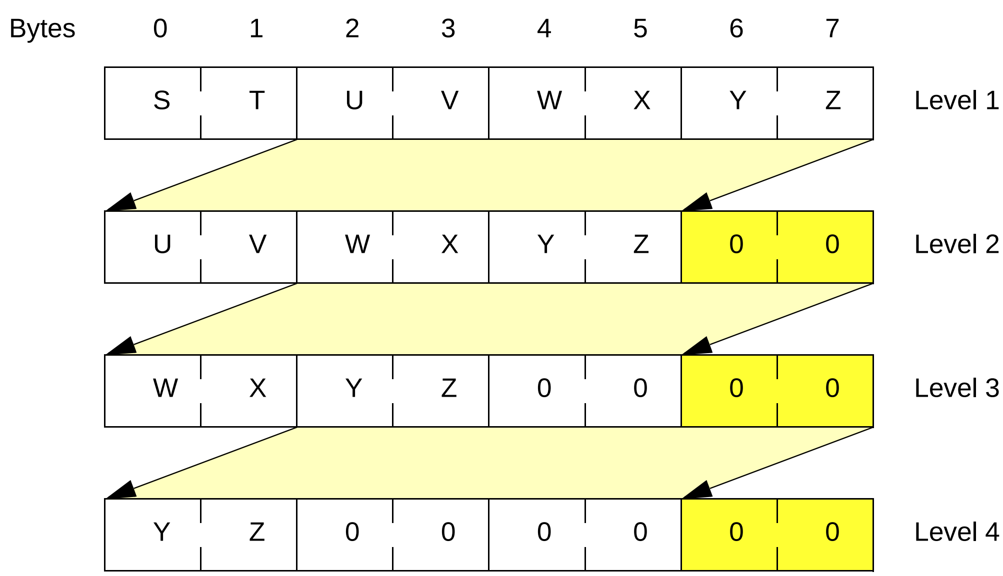
Code	Description	Reference
00b	Simple logical unit addressing method	4.7.7.1
01b	Reserved	
10b	Reserved	
11b	Extended logical unit addressing method	4.7.7.5

4.7.6.3 ~~Eight byte~~Hierarchical LUN structure

The ~~eight byte~~hierarchical LUN structure (see table 20) contains four levels of addressing fields. Each level shall use byte 0 and byte 1 to define the address and location of the SCSI target device to be addressed on that level.

If the LUN specifies that the command or task management function is to be relayed to the next level (i.e. peripheral device addressing method (see 4.7.7.2) is selected in byte 0 and byte 1 of the ~~eight byte~~hierarchical LUN structure and the BUS IDENTIFIER field is set to a value greater than zero), then the current level shall use byte 0 and byte 1 of the ~~eight byte~~hierarchical LUN structure to determine the address of the SCSI target device to which the command or task management function is to be sent. When the command or task management function is sent to the SCSI target device the ~~eight byte~~hierarchical LUN structure that was received shall be adjusted to create a new ~~eight byte~~hierarchical LUN structure (see table 19 and figure 20).

SCSI target devices shall keep track of the addressing information necessary to transmit information back through all intervening levels to the command's or task management function's originating SCSI initiator port.



A LUN may use fewer than four levels of addressing fields. When fewer than four levels of addressing fields are used, the size of the highest numbered level addressing field may be greater than two bytes (e.g., if a LUN uses three levels of addressing fields, the level three addressing field may consist of four bytes that contain EFGH).

Figure 20 — ~~Eight byte~~Hierarchical LUN structure adjustments

Table 19 — ~~Eight byte~~Hierarchical LUN 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~~hierarchical LUN structure requirements as viewed from the application client are shown in table 20.

Table 20 — Logical unit conglomerate LUN structure

Bit / Byte	7	6	5	4	3	2	1	0
0	FIRST LEVEL ADDRESSING (see table 21)							
1								
2	SECOND LEVEL ADDRESSING (see table 21)							
3								
4	THIRD LEVEL ADDRESSING (see table 21)							
5								
6	FOURTH LEVEL ADDRESSING (see table 21)							
7								

The FIRST LEVEL ADDRESSING field specifies the first level address of a SCSI target device. See table 21 for a definition of the FIRST LEVEL ADDRESSING field.

The SECOND LEVEL ADDRESSING field specifies the second level address of a SCSI target device. See table 21 for a definition of the SECOND LEVEL ADDRESSING field.

The THIRD LEVEL ADDRESSING field specifies the third level address of a SCSI target device. See table 21 for a definition of the THIRD LEVEL ADDRESSING field.

The FOURTH LEVEL ADDRESSING field specifies the fourth level address of a SCSI target device. See table 21 for a definition of the FOURTH LEVEL ADDRESSING field.

Table 21 — Format of addressing fields in the hierarchical LUN structure

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

The ADDRESS METHOD field defines the contents of the ADDRESS METHOD SPECIFIC field. See table 22 for the address methods defined for the ADDRESS METHOD field in the hierarchical LUN structure. The ADDRESS METHOD field only defines address methods for entities that are directly addressable by an application client.

Table 22 — ADDRESS METHOD field in the hierarchical LUN structure

Code	Description	Reference
00b	Peripheral device addressing method	4.7.7.2
01b	Flat space addressing method	4.7.7.3
10b	Logical unit addressing method	4.7.7.4
11b	Extended logical unit addressing method [a]	4.7.7.5

[a] Extended logical unit addresses have sizes of two bytes, four bytes, six bytes, or eight bytes. Extended logical unit addresses that are larger than two bytes shall be used only at the highest numbered level in the eight byte LUN structure (see figure 20) that is used by a LUN. Use of extended logical unit addresses shall not cause the total size of a LUN to exceed eight bytes.

4.7.7 Addressing methods

4.7.7.1 Simple logical unit addressing method

The simple logical unit addressing method (see table 23) specifies a LUN element in a logical unit conglomerate LUN structure (see 4.7.6.2).

Table 23 — Simple logical unit addressing format

Bit / Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (00b)		(MSB)					
n+1	SIMPLE LUN							(LSB)

The SIMPLE LUN field specifies a LUN element (i.e., administrative or subsidiary) in a logical unit conglomerate LUN structure.

4.7.7.2 Peripheral device addressing method

If the peripheral device addressing method (see table 24) is selected, the SCSI target device should relay the received command or task management function to the addressed dependent logical unit.

If the SCSI device does not relay any commands or task management functions to the addressed dependent logical unit, then the SCSI device shall follow the rules for ~~addressing~~processing an incorrect logical unit number described in 5.11 and 7.12.

If the SCSI device does relay some commands and task management functions to the addressed dependent logical unit, then the SCSI device shall:

a) complete any command that is not relayed with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE; and
b) terminate a task management function that is not relayed with a service response of SERVICE DELIVERY OR TARGET FAILURE.

NOTE 4 - A SCSI device may filter (i.e., not relay) commands or task management functions to prevent operations with deleterious effects from reaching a dependent logical unit (e.g., a WRITE command directed to a logical unit that is participating in a RAID volume).

Table 24 — Peripheral device addressing format

Bit / Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (00b)		BUS IDENTIFIER					
n+1	TARGET OR LUN							

The BUS IDENTIFIER field identifies the bus or path that the SCSI device shall use to relay the received command or task management function. The BUS IDENTIFIER field may use the same value encoding as the BUS NUMBER field (see 4.7.7.4) with the most significant bits set to zero. However, if the BUS IDENTIFIER field is set to 00h, then the command or task management function is to be relayed to a logical unit within the SCSI target device at the current level.

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

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

A BUS IDENTIFIER field greater than zero represents a SCSI domain that connects a group of SCSI target devices to the current level SCSI device. Each SCSI domain shall be assigned a unique bus identifier number from 1 to 63. These bus identifiers shall be used in the BUS IDENTIFIER field when assigning addresses to peripheral devices attached to the SCSI domains. When the BUS IDENTIFIER field is greater than zero, the command or task management function shall be relayed to the logical unit within the SCSI target device specified in the TARGET OR LUN field located in the SCSI domain specified by the BUS IDENTIFIER field with the LUN being set to the contents of the received LUN shifted by two bytes as described in 4.7.6.3. The SCSI target device information in the TARGET OR LUN field is a mapped representation of a target port identifier.

The SCSI target device located within the current level is addressed when the BUS IDENTIFIER field is set to zero and the TARGET OR LUN field is set to zero, also known as LUN 0 (see 4.7.4).

Figure 21 shows the selection of a logical unit using the peripheral device addressing method.

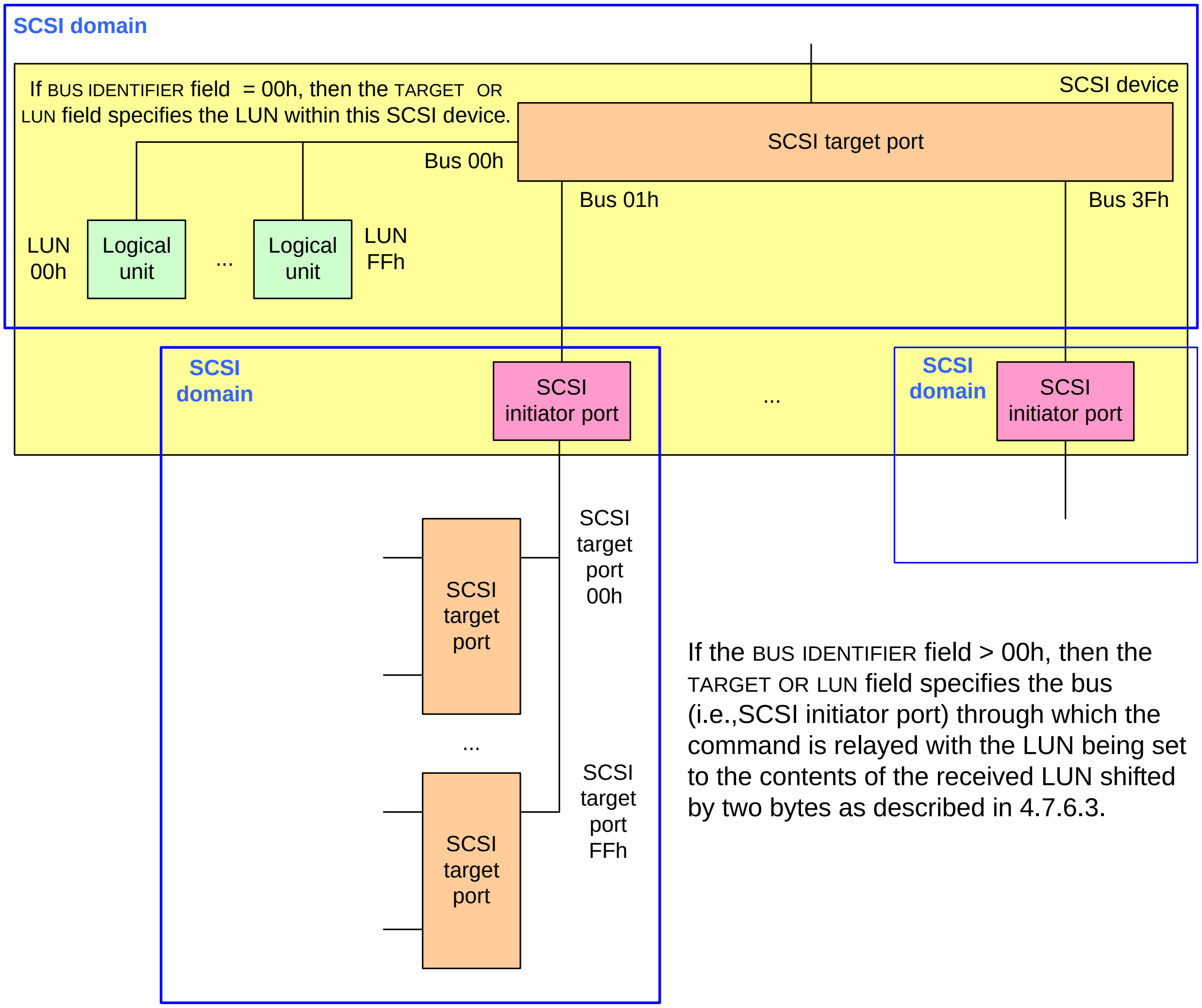


Figure 21 — Logical unit selection using the peripheral device addressing format

4.7.7.3 Flat space addressing method

The flat space addressing method (see table 25) specifies a logical unit at the current level.

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

Table 25 — Flat space addressing format

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

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

4.7.7.4 Logical unit addressing method

If the logical unit addressing method (see table 26) is selected, the SCSI target device should relay the received command or task management function to the addressed dependent logical unit.

If the SCSI device does not relay any commands or task management functions to the addressed dependent logical unit, then the SCSI device shall follow the rules for processing an incorrect logical unit number described in 5.11 and 7.12.

If the SCSI device does relay some commands and task management functions to the addressed dependent logical unit, then the SCSI device shall:

a) complete any command that is not relayed with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE; and
b) terminate a task management function that is not relayed with a service response of SERVICE DELIVERY OR TARGET FAILURE.

NOTE 5 - A SCSI device may filter (i.e., not relay) commands or task management functions to prevent operations with deleterious effects from reaching a dependent logical unit (e.g., a WRITE command directed to a logical unit that is participating in a RAID volume).

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

Table 26 — Logical unit addressing format

Bit / Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (10b)		TARGET					
n+1	BUS NUMBER			LUN				

The TARGET field, BUS NUMBER field, and LUN field address the logical unit to which the received command or task management function shall be relayed. The command or task management function shall be relayed to the logical unit specified by the LUN field within the SCSI target device specified by the TARGET field located on the bus specified by the BUS NUMBER field. The value in the LUN field shall be placed in the least significant bits of the TARGET OR LUN field in a single level LUN structure for LUNs 255 and below (see 4.7.5). The TARGET field contains a mapped representation of a target port identifier.

Figure 22 shows the selection of a logical unit using the logical unit addressing method.

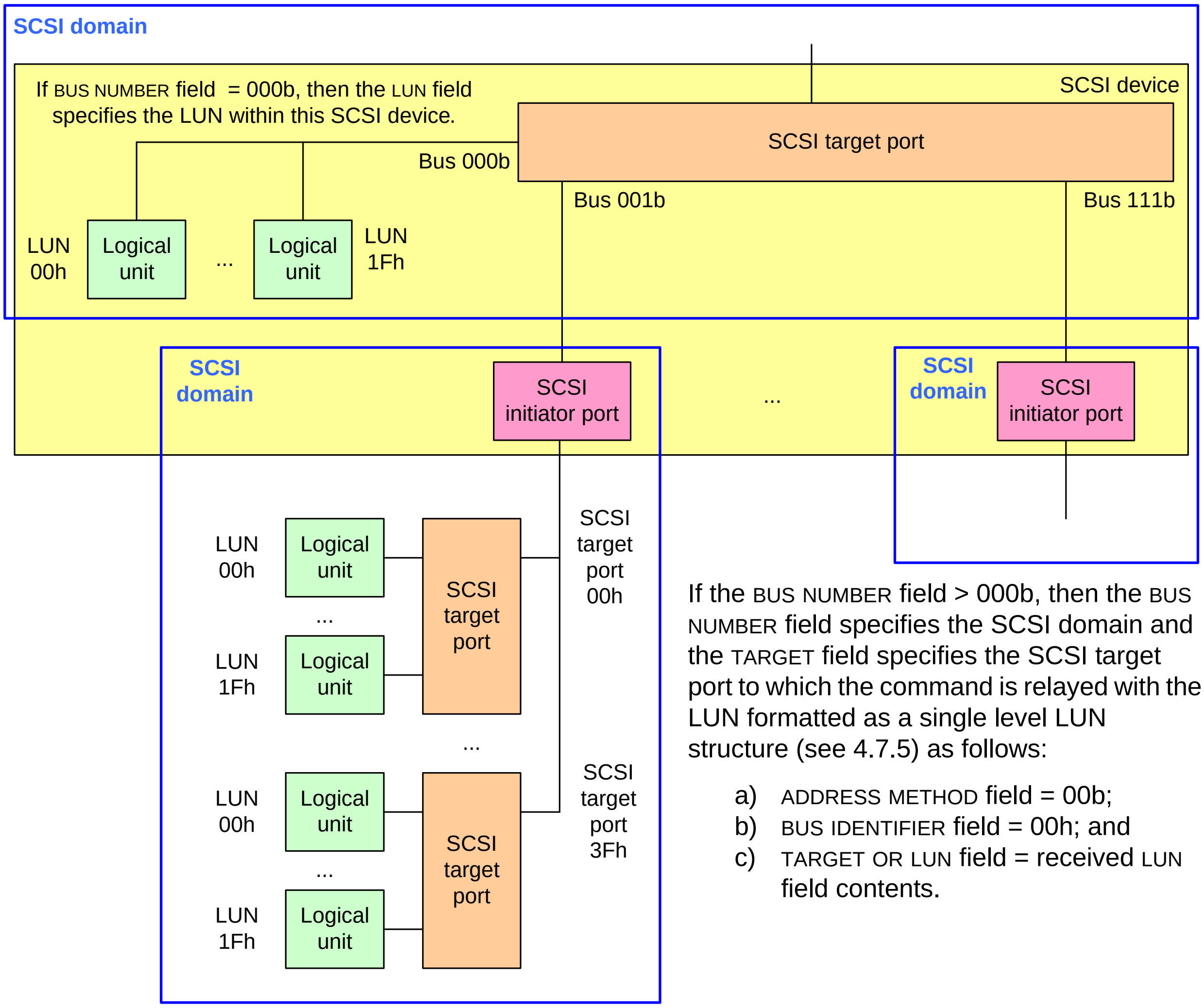


Figure 22 — Logical unit selection using the logical unit addressing format

4.7.7.5 Extended logical unit addressing

Extended logical unit addressing (see table 27) specifies a logical unit at the current level.

Extended logical unit addressing builds on the formats defined for dependent logical units (see 4.6.17.4) but may be used by SCSI devices having single level logical unit structure or multiple level logical unit structure. In dependent logical unit addressing, the logical unit information at each level, except the highest numbered level used, fits in exactly two bytes. Extended logical unit addresses have sizes of two bytes, four bytes, six bytes, or eight bytes. Use of extended logical unit addresses shall not cause the total size of a LUN to exceed eight bytes.

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

Extended logical units are identified by the ADDRESS METHOD field (see table 22 in 4.7.6.3) in the same manner as is the case for dependent logical units. An ADDRESS METHOD field value of 11b specifies the extended logical unit addressing method.

Table 27 — Extended logical unit addressing format

Bit / Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH		EXTENDED ADDRESS METHOD			
n+1	(MSB)							
•••	EXTENDED ADDRESS METHOD SPECIFIC							
m								(LSB)

The LENGTH field (see table 28) specifies the length of the EXTENDED ADDRESS METHOD SPECIFIC field. A LUN that includes a LENGTH field value that goes beyond the LUN field length supported by the transport protocol is invalid and shall follow the rules for processing an incorrect logical unit number described in 5.11.

Table 28 — LENGTH field and related sizes

Code	Size in bytes of		Reference
	EXTENDED ADDRESS METHOD SPECIFIC field	Extended logical unit addressing format	
00b	1	2	table 29
01b	3	4	table 30
10b	5	6	table 31
11b	7	8	table 32

Table 29, table 30, table 31, and table 32 show the four extended logical unit addressing formats.

Table 29 — Two byte extended logical unit addressing format

Bit / Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (00b)		EXTENDED ADDRESS METHOD			
n+1	EXTENDED ADDRESS METHOD SPECIFIC							

Table 30 — Four byte extended logical unit addressing format

Bit / Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (01b)		EXTENDED ADDRESS METHOD			
n+1	EXTENDED ADDRESS METHOD SPECIFIC							
•••								
n+3								

Table 31 — Six byte extended logical unit addressing format

Bit / Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (10b)		EXTENDED ADDRESS METHOD			
n+1	EXTENDED ADDRESS METHOD SPECIFIC							
•••								
n+5								

Table 32 — Eight byte extended logical unit addressing format

Bit / Byte	7	6	5	4	3	2	1	0
0	ADDRESS METHOD (11b)		LENGTH (11b)		EXTENDED ADDRESS METHOD			
1	EXTENDED ADDRESS METHOD SPECIFIC							
•••								
7								

The EXTENDED ADDRESS METHOD field combined with the LENGTH field (see table 33) specifies the type and size of extended logical unit address found in the EXTENDED ADDRESS METHOD SPECIFIC field.

Table 33 — Logical unit extended addressing

EXTENDED ADDRESS METHOD **Codes**	LENGTH **Code(s)**	**Description**	**Hierarchical LUN usage [a]**	**Logical unit conglomerate LUN usage [b]**	**Reference**
0h	00b to 11b	Reserved			
1h	00b	Well known logical unit	yes	no	4.7.7.5.1
	01b to 11b	Reserved			
2h	01b	Extended flat space addressing	yes	yes	4.7.7.5.2
	10b	Long extended flat space addressing	yes	yes	4.7.7.5.3
	00b, 11b	Reserved			
3h - Dh	00b to 11b	Reserved			
Eh	00b to 10b	Reserved			
	11b	Restricted for FC-SB-4	no	no	FC-SB-4
Fh	00b to 10b	Reserved			
	11b	Logical unit not specified	yes	no	4.7.7.5.4

a Key:
yes = Other subclauses in this standard are allowed the use of a specific logical unit extended addressing method in hierarchical LUN structures (see 4.7.6.3).
no = Only a single level LUN structure allowed (see 4.7.5).

b Key:
yes = Other subclauses in this standard are allowed the use of a specific logical unit extended addressing method in logical unit conglomerate LUN structures (see 4.7.6.2).
no = Other subclauses in this standard are not allowed the use of a specific logical unit extended addressing method in logical unit conglomerate LUN structures.

4.7.7.5.1 Well known logical unit addressing

A SCSI target device may support zero or more well known logical units (see 4.6.25). A single SCSI target device shall only support one instance of each supported well known logical unit. All well known logical units within a SCSI target device shall be accessible from all SCSI target ports contained within the SCSI target device.

Well known logical units are addressed using the well known logical unit extended address format (see table 34).

Table 34 — Well known logical unit extended addressing format

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

The W-LUN field specifies the well known logical unit to be addressed (see SPC-4).

4.7.7.5.2 Extended flat space addressing method

The extended flat space addressing method (see table 35) specifies a logical unit at the current level or in a conglomerate logical unit.

The contents of all hierarchical structure addressing fields following an extended flat space addressing method addressing field shall be ignored. The contents of all logical unit conglomerate LUN structure elements following an extended flat space addressing method addressing field shall be processed as described in 4.7.6.2.

Table 35 — Extended flat space addressing format

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

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

4.7.7.5.3 Long extended flat space addressing method

The long extended flat space addressing method (see table 36) specifies a logical unit at the current level.

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

Table 36 — Long extended flat space addressing format

Bit / Byte	7	6	5	4	3	2	1	0
n	ADDRESS METHOD (11b)		LENGTH (10b)		EXTENDED ADDRESS METHOD (2h)			
n+1	(MSB)							
•••	LONG EXTENDED FLAT SPACE LUN							
n+5								(LSB)

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

4.7.7.5.4 Logical unit not specified addressing

Logical unit not specified addressing (see table 37) shall be used to indicate that no logical unit of any kind is specified.

Table 37 — Logical unit not specified extended addressing format

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

4.8 Nexus

4.8.1 Nexus overview

The nexus represents a relationship between a SCSI initiator port, a SCSI target port, optionally a logical unit, and optionally a command. The notations for the types of nexuses are:

a) I_T nexus;
b) I_T_L nexus;
c) I_T_L_Q nexus; and
d) I_T_L_x nexus.

Table 38 defines the types of nexuses and the identifiers used to construct each of them.

Table 38 — Nexus

Nexus [a]	Identifiers used to construct nexuses	Reference
I_T nexus	Initiator port identifier Target port identifier	4.6.8.2 4.6.6.2
I_T_L nexus	Initiator port identifier Target port identifier LUN	4.6.8.2 4.6.6.2 4.6.17.2
I_T_L_Q nexus	Initiator port identifier Target port identifier LUN Command identifier	4.6.8.2 4.6.6.2 4.6.17.2 4.8.2
[a] I_T_L_x nexus specifies either an I_T_L nexus or an I_T_L_Q nexus.		

4.8.2 Command identifier

A command identifier (i.e., the Q in an I_T_L_Q nexus) is assigned by a SCSI initiator device to uniquely identify one command in the context of a particular I_T_L nexus, allowing more than one command to be outstanding for that I_T_L nexus at the same time. Each SCSI transport protocol defines the size of the command identifier, up to a maximum of 64 bits, to be used by SCSI ports that support that SCSI transport protocol.

SCSI transport protocols may define additional restrictions on command identifier assignments (e.g., requiring command identifiers to be unique per I_T nexus or per I_T_L nexus, or sharing command identifier values with other uses such as task management functions).

4.8.3 Nexus usage rules

An I_T_L_Q nexus that is in use (i.e., during the interval bounded by the events specified in 5.5) shall be unique as seen by the SCSI initiator port originating the command and the logical unit to which the command was addressed, otherwise an overlapped command condition exists (see 5.10). An I_T_L_Q nexus is unique if one or more of its components is unique within the specified time interval.

The SCSI initiator device shall not create more than one command from a specific SCSI initiator port having identical values for the target port identifier, LUN, and command identifier.

4.9 SCSI ports

4.9.1 SCSI port configurations

A SCSI device contains only the following combinations of SCSI ports:

a) all SCSI target ports;
b) all SCSI initiator ports; or
c) any combination of SCSI target ports and SCSI initiator ports.

Some of the SCSI port configurations possible for a SCSI device are shown in figure 23.

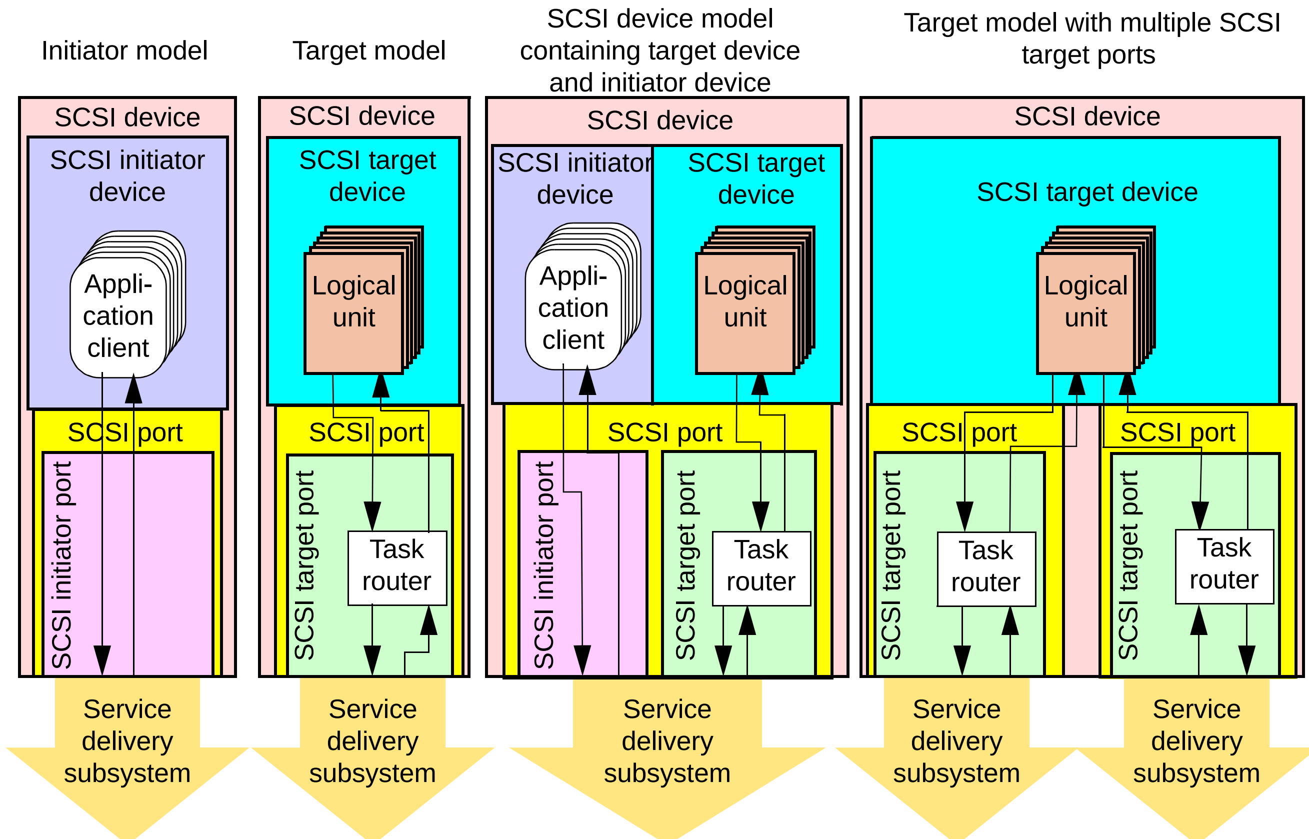


Figure 23 — SCSI device functional models

Additional SCSI initiator ports and SCSI target ports allow the definition of multiple I_T nexuses through which the application client may access the device server.

4.9.2 SCSI devices with multiple SCSI ports

The model for a SCSI device with multiple SCSI ports is a single SCSI device (see 4.6.4) containing either:

a) SCSI target device (see 4.6.9) ~~with~~and multiple SCSI ~~target~~ ports (see 4.6.5) with each SCSI port containing a SCSI target port (see 4.6.6);
b) SCSI initiator device (see 4.6.26) ~~with~~and multiple SCSI ~~initiator~~ ports, with each SCSI port containing a SCSI initiator port (see 4.6.8); or
c) ~~SCSI device containing~~ a SCSI initiator device, ~~and~~ a SCSI target device, and multiple SCSI ports with each SCSI port containing a SCSI target port and/or SCSI initiator port.

The identifiers representing the SCSI ports shall meet the requirements for initiator port identifiers (see 4.6.8.2) or target port identifiers (see 4.6.6.2). How a SCSI device with multiple SCSI ports is viewed by counterpart SCSI devices in the SCSI domain also depends on whether a SCSI initiator port is examining a SCSI target port, or a SCSI target port is servicing a SCSI initiator port.

If one SCSI target port is being used by a SCSI initiator port, accesses attempted through other SCSI target ports may:

a) receive a status of BUSY (see 5.3.1); or
b) be accepted as if the other SCSI target ports were not in use.

4.9.3 SCSI target device with multiple SCSI ports structure

Figure 24 shows the structure of a SCSI target device with multiple SCSI ports each containing a SCSI target port. Each SCSI target port contains a task router that is shared by a collection of logical units. Each logical unit contains a single task manager and a single device server.

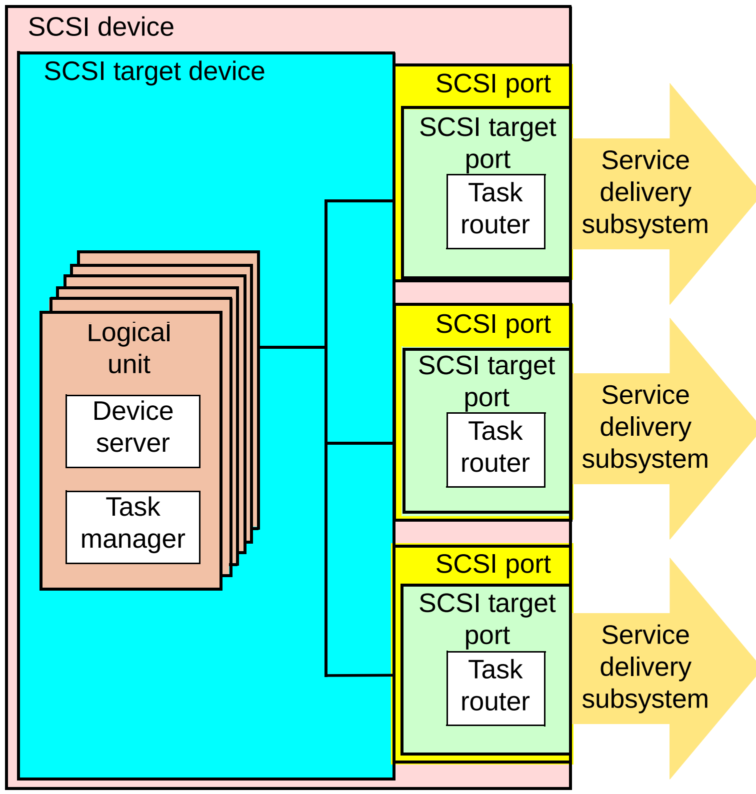


Figure 24 — SCSI device with multiple SCSI target ports structure model

Each device server shall indicate the presence of multiple SCSI target ports by setting the MULTIP bit to one in its standard INQUIRY data (see SPC-4).

Two-way communications shall be possible between all logical units and all SCSI target ports in a SCSI device. However, communications between any logical unit and any SCSI target port in a SCSI device may be inactive. Two-way communications shall be available between each task manager and all task routers in the SCSI target ports in the SCSI device. Each SCSI target port in a SCSI device shall accept commands sent to LUN 0 or the REPORT LUNS well known logical unit, and the task router in that SCSI target port shall route the commands to a device server in a logical unit in the SCSI device for processing. REPORT LUNS commands (see SPC-4) shall be accepted by the logical unit with the LUN zero or the REPORT LUNS well known logical unit from any SCSI target port in the SCSI device, and the logical unit shall return the logical unit inventory available via that SCSI target port. An application client determines the availability of the same logical unit through multiple SCSI target ports in a SCSI device by matching logical unit name values in the Device Identification VPD page (see SPC-4).

4.9.4 SCSI initiator device with multiple SCSI initiator ports structure

Figure 25 shows the structure of a SCSI initiator device with multiple SCSI ports each containing a SCSI initiator port. Each SCSI initiator port is shared by a collection of application clients.

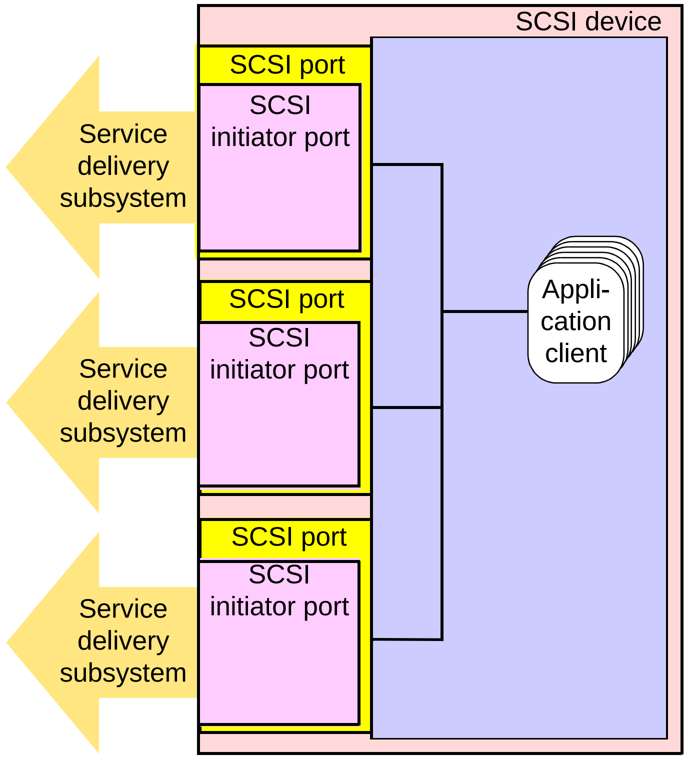


Figure 25 — SCSI initiator device with multiple SCSI initiator ports structure model

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 SCSI initiator 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).

4.9.5 SCSI device with multiple SCSI ports structure

Figure 26 shows the structure of a SCSI device containing a SCSI target device and a SCSI initiator device, and multiple SCSI ports. Each SCSI port contains a SCSI target port and a SCSI initiator port. This SCSI device may also contain SCSI ports that only contain a SCSI target port or a SCSI initiator port. Each SCSI port consists of a SCSI target port containing a task router and a SCSI initiator port and is shared by a collection of logical units and application clients. Each logical unit contains a task manager and a device server.

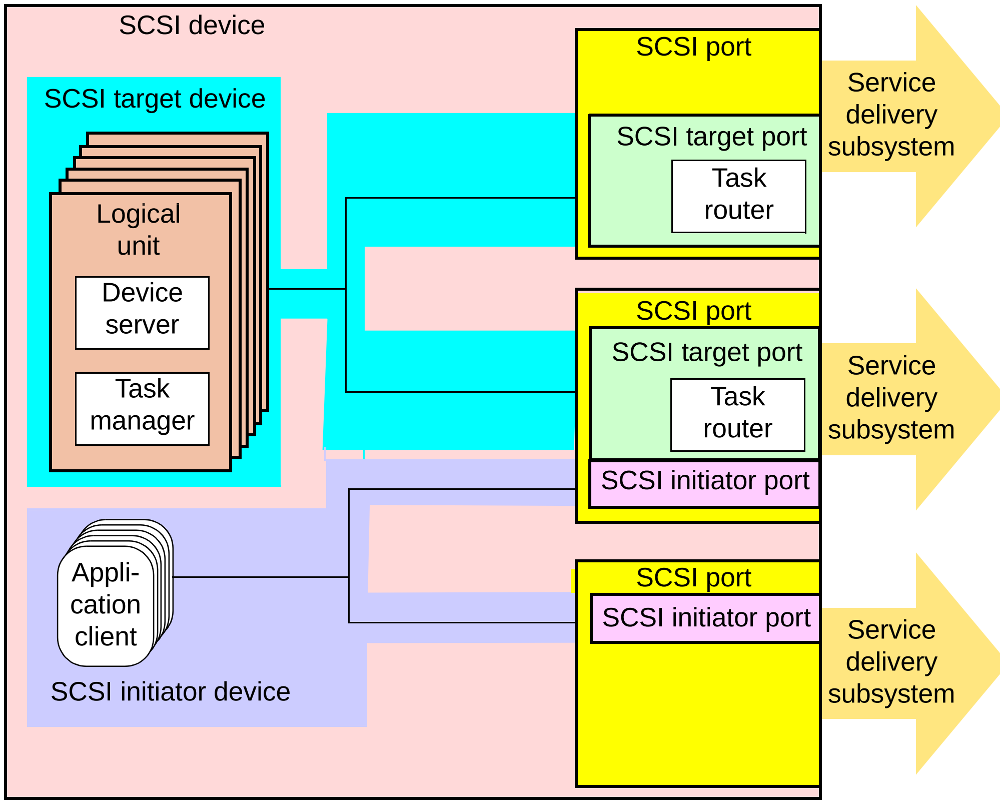


Figure 26 — SCSI device with multiple SCSI ports structure model

Two-way communications shall be possible between all logical units and all SCSI target ports in a SCSI device. However, communications between any logical unit and any SCSI target port in a SCSI device may be inactive. Two-way communications shall be available between each task manager and all task routers in the SCSI target ports in the SCSI device. Each SCSI target port in a SCSI device shall accept commands sent to LUN 0 or the REPORT LUNS well known logical unit, and the task router in that SCSI target port shall route the commands to a device server in a logical unit in the SCSI device for processing. REPORT LUNS commands (see SPC-4) shall be accepted by the logical unit with the LUN zero or the REPORT LUNS well known logical unit from any SCSI target port in the SCSI device, and the logical unit shall return the logical unit inventory available via that SCSI target port. An application client determines the availability of the same logical unit through multiple SCSI target ports in a SCSI device by matching logical unit name values in the Device Identification VPD page (see SPC-4).

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

4.9.6 SCSI initiator device view of SCSI target device with multiple SCSI target ports

A SCSI target device may have SCSI target ports connected to different SCSI domains such that a SCSI initiator port is only able to communicate with the logical units in the SCSI target device using the SCSI target ports in a single SCSI domain. 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 27 and figure 28 show two examples of such configurations.

Figure 27 shows a SCSI target device with multiple SCSI ports each containing a SCSI target port participating in a single SCSI domain with two SCSI initiator devices. There are three SCSI devices, one of which has two SCSI target ports, and two of which have one SCSI initiator port each. There are two target port identifiers and two initiator port identifiers in this SCSI domain. Using the INQUIRY command Device Identification VPD page (see

SPC-4), the application clients in each of the SCSI initiator devices have the ability to discover if the logical units in the SCSI target devices are accessible via multiple SCSI target ports and map the configuration of the SCSI target device.

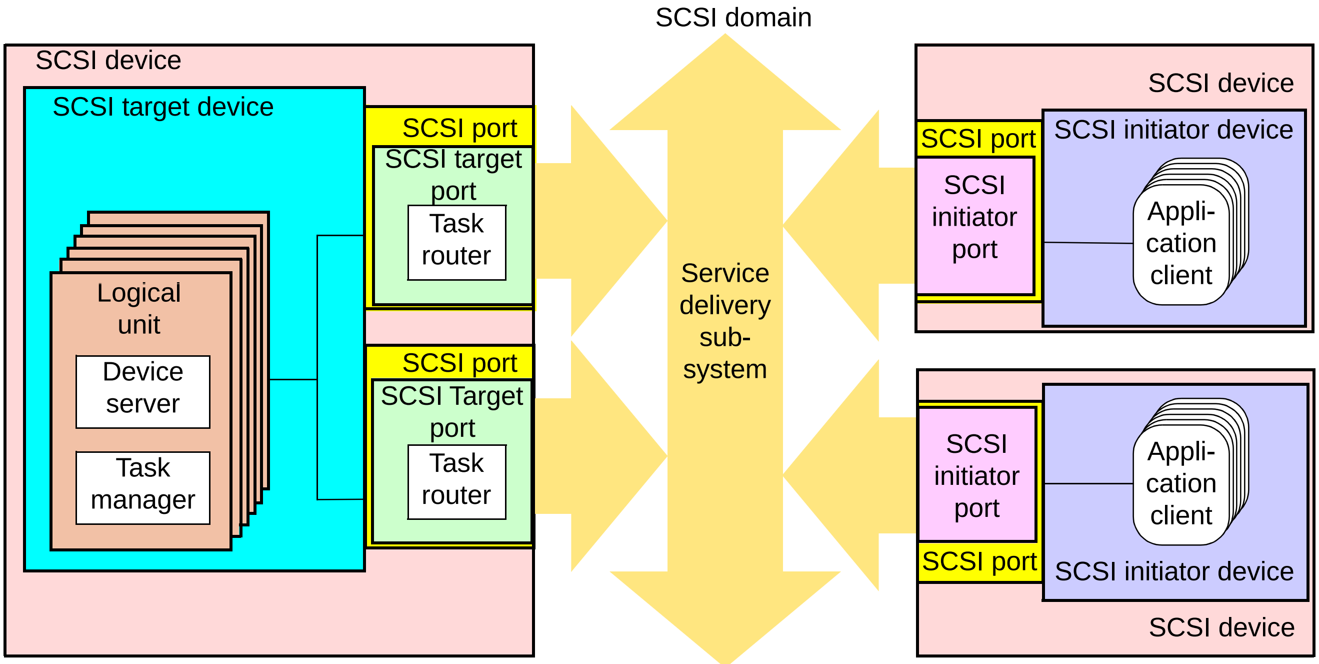


Figure 27 — SCSI target device configured in a single SCSI domain

Figure 28 shows a SCSI target device with multiple SCSI ports each containing a SCSI target port participating in two SCSI domains and a SCSI initiator device with multiple SCSI ports each containing a SCSI initiator port participating in the same two SCSI domains. There is one SCSI target device with two SCSI target ports and one SCSI initiator device with two SCSI initiator ports. There is one target port identifier and one initiator port identifier in each of the two SCSI domains. Using the INQUIRY command Device Identification VPD page (see SPC-4), the application clients in the SCSI initiator device have the ability to discover that logical units in the SCSI target device are accessible via multiple SCSI initiator ports and multiple SCSI target ports and map the configuration. However, application clients may not be able to distinguish between the configuration shown in figure 28 and the configuration shown in figure 29.

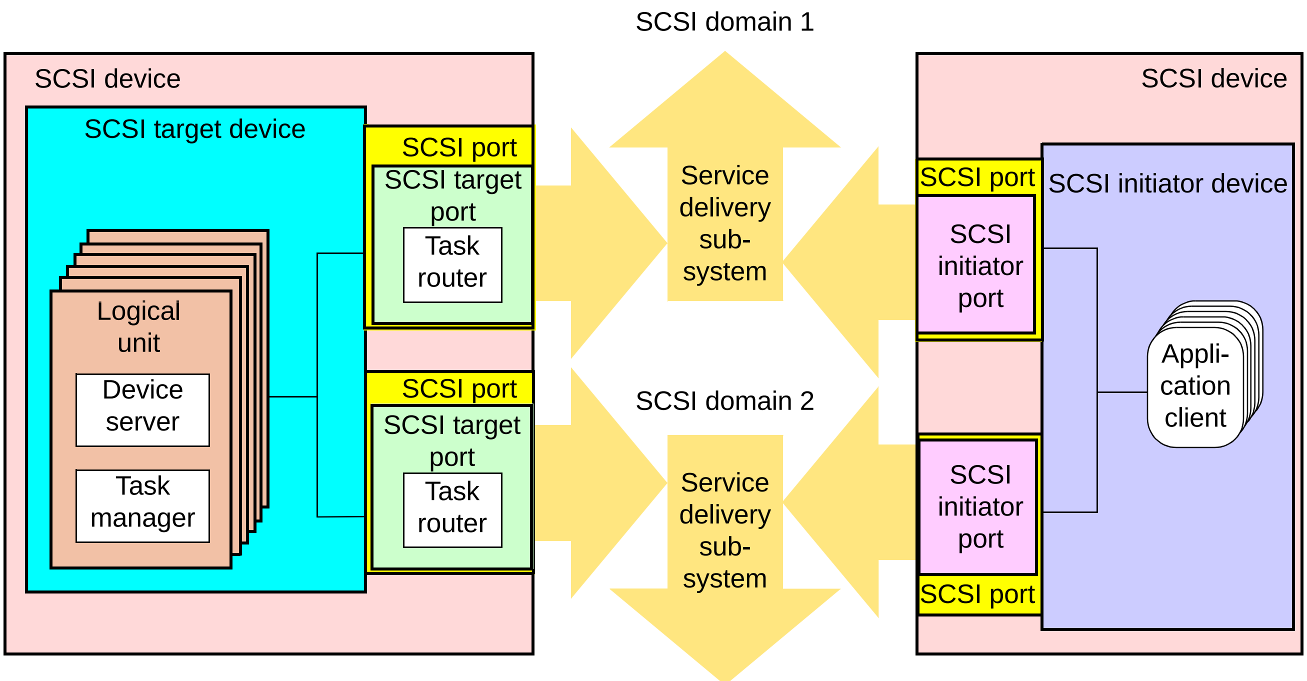


Figure 28 — SCSI target device configured in multiple SCSI domains

Figure 29 shows the same configuration as figure 28 except that the two SCSI domains have been replaced by a single SCSI domain.

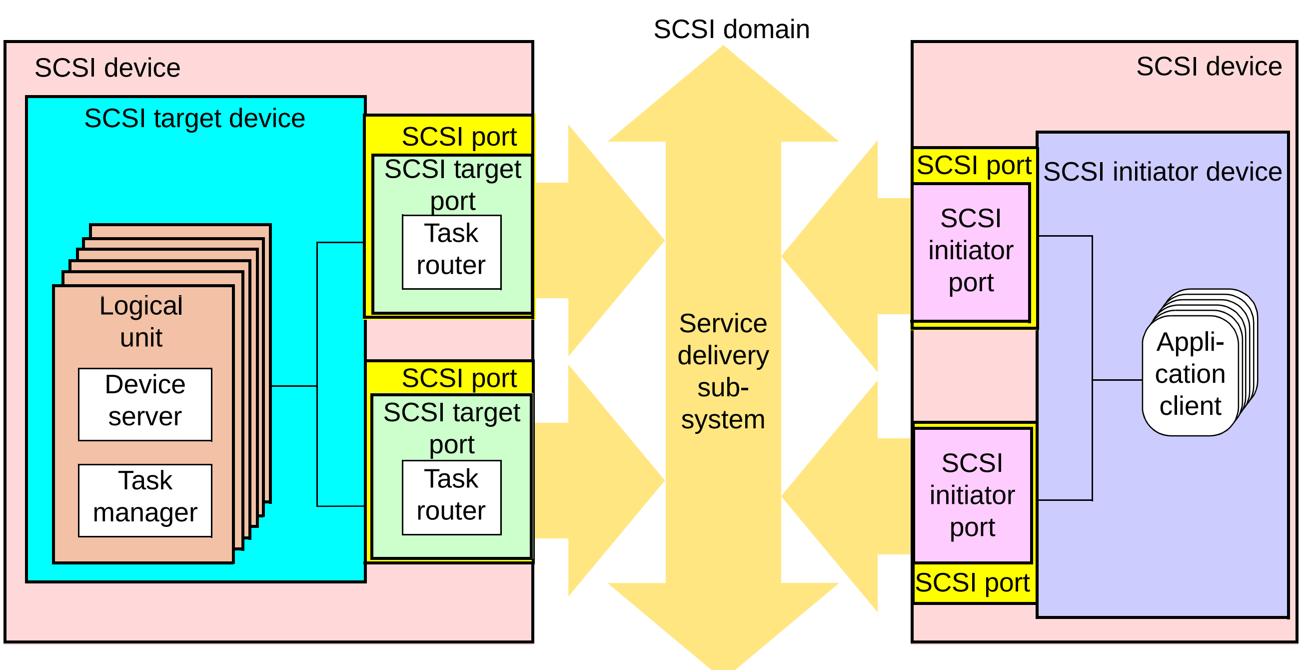


Figure 29 — SCSI target device and SCSI initiator device configured in a single SCSI domain

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

4.9.7 SCSI target device view of a SCSI initiator device with multiple SCSI initiator ports

This standard does not require a SCSI target device to be able to detect that a SCSI initiator device contains more than one SCSI initiator port. In the cases where a SCSI target device does not detect that a SCSI initiator device contains more than one SCSI initiator port, the SCSI target device interacts with the SCSI initiator device as if each SCSI initiator port was contained in a separate SCSI initiator device (e.g., a SCSI target device operates in the configurations shown in figure 28 and figure 29 in the same way it operates in the configuration shown in figure 27).

NOTE 6 - The implications of this view of a SCSI initiator device are more far reaching than are immediately apparent (e.g., after a SCSI initiator device makes a persistent exclusive access reservation via one SCSI initiator port, access is denied to the other SCSI initiator port(s) on that same SCSI initiator device).

4.10 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 30.

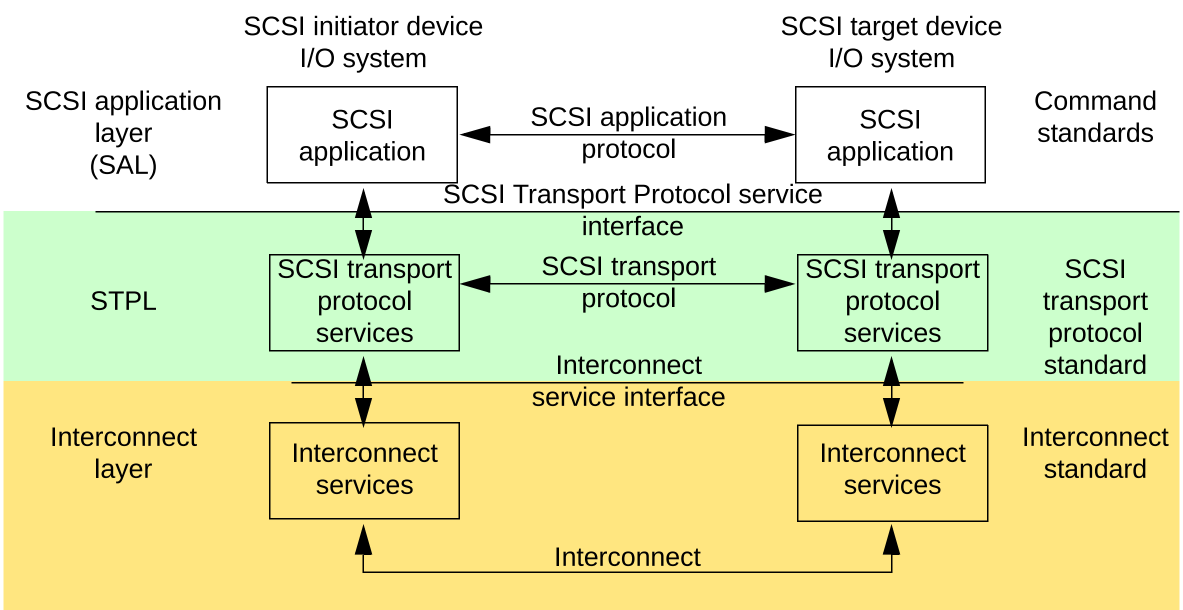


Figure 30 — Protocol service reference model

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

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

The set of SCSI transport protocol services implemented by a service delivery subsystem identify external behavioral requirements that apply to SCSI transport protocol standards. While these SCSI transport protocol services may serve as a guide for designing reusable software or firmware that is adaptable to different SCSI transport protocols, there is no requirement for an implementation to provide the service interfaces specified in this standard.

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

Interactions between the SAL and STPL are defined with respect to the SAL and may originate in either layer. An outgoing interaction is modeled as a procedure call invoking an STPL service (e.g., invoking an operation defined by the SCSI Target Port class or the SCSI Initiator Port class). An incoming interaction is modeled as a procedure call invoked by the STPL (e.g., invoking an operation defined by the Application Client class, the Device Server class, or the Task Manager class).

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

The following types of service interactions between layers are defined:

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

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

Figure 31 shows the relationships between the four SCSI transport protocol service types.

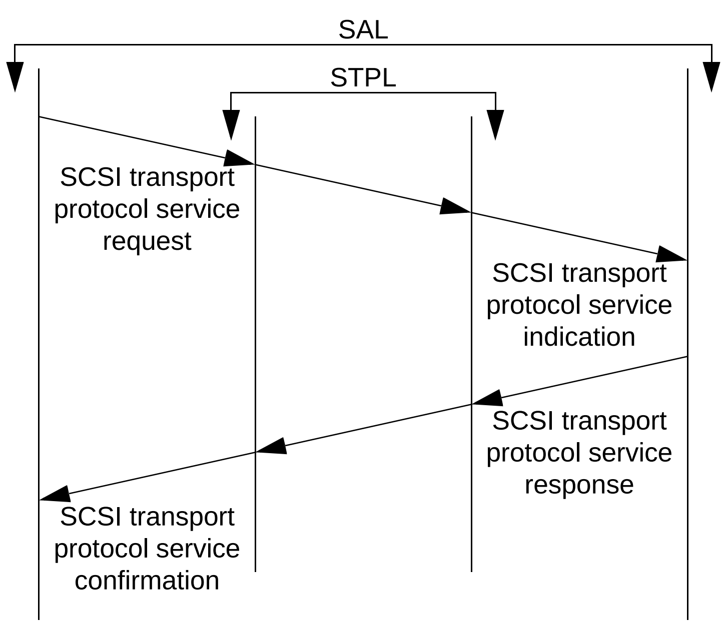


Figure 31 — SCSI transport protocol service mode

Figure 32 shows how SCSI transport protocol services may be used to process a client-server request-response transaction at the SAL.

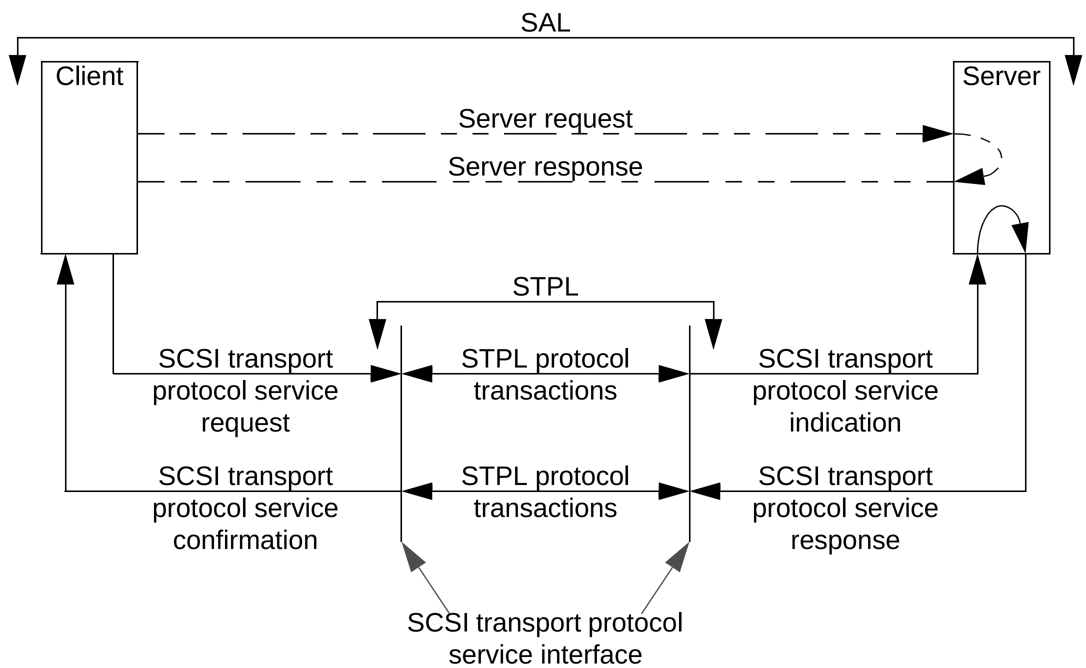


Figure 32 — Request-Response SAL transaction and related STPL services

The dashed lines in figure 32 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 32 show the corresponding SCSI transport protocol services and STPL transactions that are used to transport the data.

When a device server invokes a data transfer SCSI transport protocol service, the interactions required to transfer the data do not involve the application client. Only the STPL in the SCSI device that also contains the application client is involved. Figure 33 shows the relationships between the SCSI transport protocol service types involved in a data transfer request.

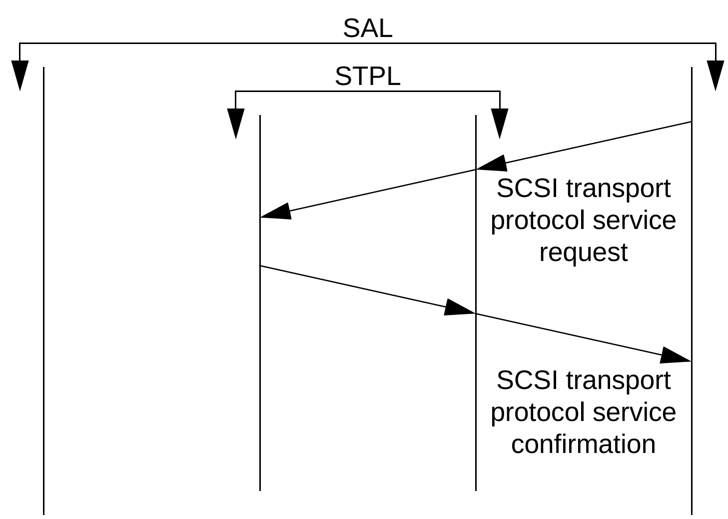
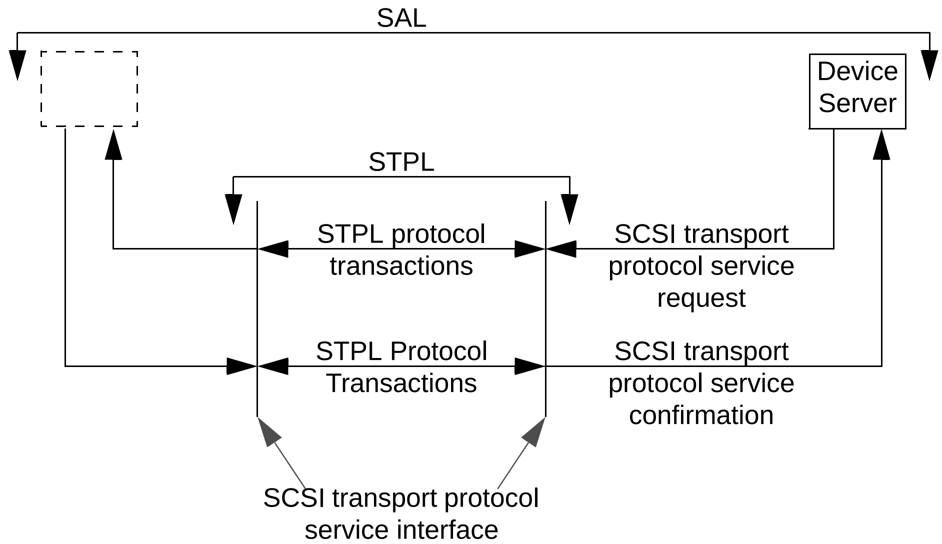


Figure 33 — SCSI transport protocol service model for data transfers

Figure 34 shows how SCSI transport protocol services may be used to process a device server data transfer transaction.



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

Figure 34 — Device server data transfer transaction and related STPL services

When a device server invokes a Terminate Data Transfer SCSI transport protocol service, the interactions required to complete the service do not involve the SCSI transport protocol service interface or the application client. Only the STPL in the SCSI device that also contains the device server is involved. Figure 35 shows the relationships between the SCSI transport protocol service types involved in a Terminate Data Transfer request.

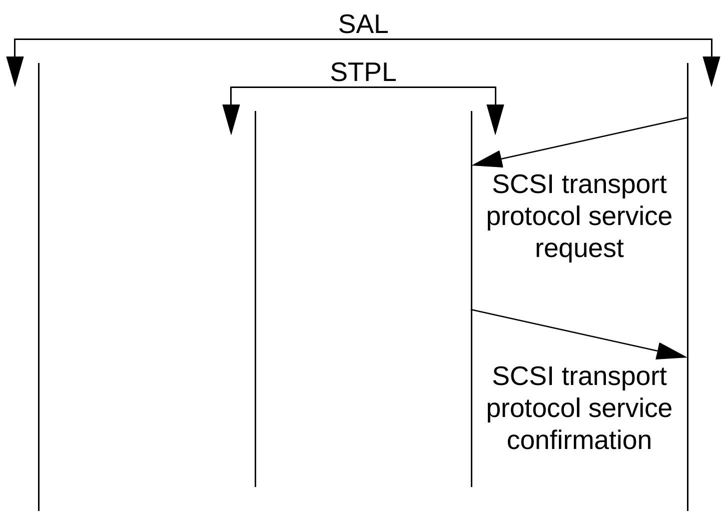


Figure 35 — SCSI transport protocol service model for Terminate Data Transfer

Figure 36 shows how SCSI transport protocol services may be used to process a device server Terminate Data Transfer transaction.

.

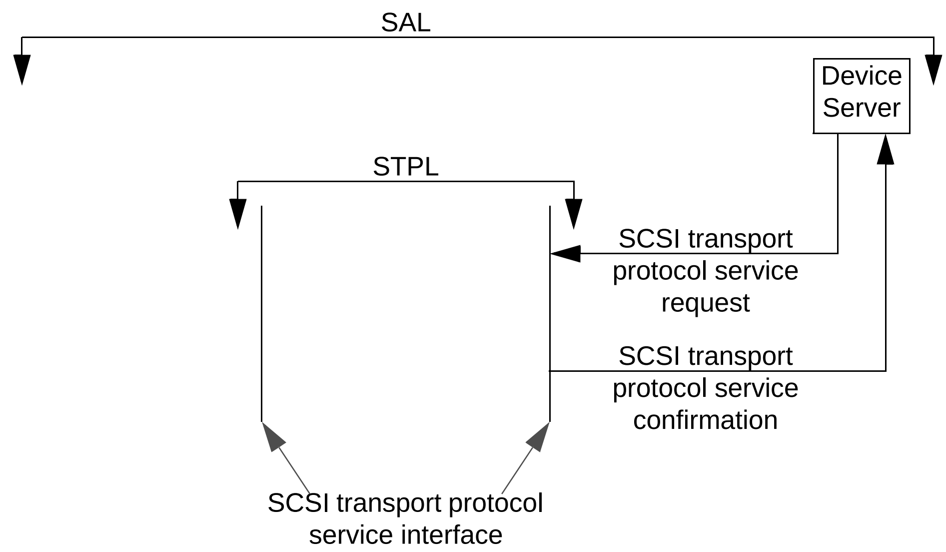


Figure 36 — Device server Terminate Data Transfer transaction and related STPL services

5 SCSI command model

5.1 The Execute Command procedure call

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

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

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the command (see 4.8).

CDB: Command descriptor block (see 5.2).

Task Attribute: A value specifying one of the task attributes defined in 8.6.

Data-In Buffer Size: The number of bytes available for data transfers to the Data-In Buffer (see 5.4.3). SCSI transport protocols may interpret the Data-In Buffer Size to include both the size and the location of the Data-In Buffer.

Data-Out Buffer: A buffer (see 5.4.3) containing command specific information to be sent to the logical unit (e.g., data or parameter lists needed to process the command). The buffer size is indicated by the Data-Out Buffer Size argument. The content of the buffer shall not change during the lifetime of the command (see 5.5) as viewed by the application client.

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

CRN: When the CRN is used, all commands on an I_T_L nexus shall include a CRN argument that is incremented by one. The CRN shall be set to one for each I_T_L nexus involving the SCSI port after the SCSI port receives a hard reset or detects I_T nexus loss. The CRN shall be set to one after it reaches the maximum CRN value supported by the protocol. The CRN value zero shall be reserved for use as defined by the SCSI transport protocol. It is not an error for the application client to provide a CRN when CRN is not supported by the SCSI transport protocol or logical unit. See the SCSI transport protocol standards for rules regarding CRN checking.

Command Priority: The priority assigned to the command (see 8.7).

Output arguments:

Data-In Buffer: A buffer (see 5.4.3) to contain command specific information returned by the logical unit by the time of command completion. The **Execute Command** procedure call shall not return GOOD status or CONDITION MET status unless the buffer contents are valid. The application client shall treat the buffer contents as invalid unless **Execute Command** procedure call returns GOOD status or CONDITION MET status. While some valid data may be present for other values of status, the application client should rely on additional information from the logical unit (e.g., sense data) to determine the state of the buffer contents. If the command terminates with a service response of SERVICE DELIVERY OR TARGET FAILURE, the application client shall consider the buffer to be undefined.

Sense Data: A buffer containing sense data returned in the same I_T_L_Q nexus transaction as status (see 5.13). The buffer length is indicated by the Sense Data Length argument. If the command terminates with a service response of SERVICE DELIVERY OR TARGET FAILURE, the application client shall consider the sense data to be undefined.

Sense Data Length: The length in bytes of the Sense Data (see 5.13).

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

Status Qualifier: Additional information about the indicated command completion status (see 5.3.2).

One of the following SCSI transport protocol specific service responses shall be returned:

COMMAND COMPLETE: A logical unit response indicating that the command has completed. The Status argument shall have one of the values specified in 5.3.

SERVICE DELIVERY OR TARGET FAILURE: The command has been terminated due to a service delivery failure (see 3.1.116) or SCSI target device malfunction. All output arguments are invalid.

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

5.2 Command descriptor block (CDB)

The CDB defines the operation to be performed by the device server. See SPC-4 for the CDB formats.

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

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

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

All CDBs shall contain a CONTROL byte (see table 39). The location of the CONTROL byte within a CDB depends on the CDB format (see SPC-4).

Table 39 — CONTROL byte

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

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

The NACA (Normal ACA) bit specifies whether an auto contingent allegiance (ACA) is established if the command terminates with CHECK CONDITION status. A NACA bit set to one specifies that an ACA shall be established. A NACA bit set to zero specifies that an ACA shall not be established. The actions for ACA are specified in 5.9. Actions that may be required when an ACA is not established are described in 5.8. All logical units shall implement support for the NACA value of zero and may support the NACA value of one (i.e., ACA). The ability to support a NACA value of one is indicated with the NORMACA bit in the standard INQUIRY data (see SPC-4).

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

5.3 Status

5.3.1 Status codes

The status codes are specified in table 40. Status shall be sent from the device server to the application client whenever a command completes with a service response of COMMAND COMPLETE.

Table 40 — Status codes

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

Sense data may be delivered in the buffer defined by the Sense Data argument of the Execute Command procedure call (see 5.1) for any status code.

Definitions for each status code are as follows:

GOOD. This status indicates that the device server has completed the command without error.

CHECK CONDITION. This status indicates that sense data has been delivered in the buffer defined by the Sense Data argument to the **Execute Command** procedure call (see 5.13). Additional actions that are required when CHECK CONDITION status is returned are described in 5.8.

CONDITION MET. The use of this status is limited to commands for which it is specified (see the PRE-FETCH commands in SBC-3).

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 through the target port on which the status is returned and zero or more other target ports. 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-4), termination of a command with BUSY status shall cause a unit attention condition to be established for the SCSI initiator port on the I_T nexus that sent the command with an additional sense code set to PREVIOUS BUSY STATUS.

The status qualifier, when supported by a SCSI transport protocol, may provide the SCSI initiator port with more information about when the command should be retransmitted (see 5.3.2).

RESERVATION CONFLICT. This status shall be returned whenever a command is sent by an application client to a logical unit in a way that conflicts with an existing reservation. (See the PERSISTENT RESERVE OUT command and PERSISTENT RESERVE IN command in SPC-4.)

If the UA_INTLCK_CTRL field in the Control mode page contains 11b (see SPC-4), termination of a command with RESERVATION CONFLICT status shall cause a unit attention condition to be established for the SCSI initiator port on the I_T nexus that sent the command with an additional sense code set to PREVIOUS RESERVATION CONFLICT STATUS.

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

The logical unit should allow at least one command in the task set for each supported I_T nexus (i.e., a logical unit should allow at least one command into the task set for each I_T nexus that has been identified in a SCSI transport protocol specific manner (e.g., a login), or by the successful reception of a command).

The status qualifier, when supported by a SCSI transport protocol, may provide the SCSI initiator port with more information about when the command should be retransmitted (see 5.3.2).

If the UA_INTLCK_CTRL field in the Control mode page contains 11b (see SPC-4), termination of a command with TASK SET FULL status shall cause a unit attention condition to be established for the SCSI initiator port on the I_T nexus that sent the command with an additional sense code set to PREVIOUS TASK SET FULL STATUS.

ACA ACTIVE. This status shall be returned as described in 5.9.2 and 5.9.3 when an ACA exists within a task set. The application client may reissue the command on the same I_T nexus after the ACA condition has been cleared.

TASK ABORTED. This status shall be returned when a command is aborted by a command or task management function on another I_T nexus and the Control mode page TAS bit is set to one (see 5.6).

5.3.2 Status qualifier

The status qualifier provides additional information about the reason for the status code (see 5.3.1).

The status qualifier format is as shown in table 41.

Table 41 — Status qualifier format

<table>
<tr><th>Bit
Byte</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>0</td><td colspan="2">SCOPE</td><td colspan="6">(MSB)
QUALIFIER</td></tr>
<tr><td>1</td><td colspan="8">(LSB)</td></tr>
</table>

The SCOPE field (see table 42) indicates the logical unit(s) to which the status qualifier applies.

Table 42 — SCOPE field

Code	Affected logical unit(s)	Affected nexus(es)
00b	The logical unit addressed by the command associated with the status.	All I_T nexus(es).
01b	All logical units accessible by the SCSI target port through which the command associated with the status was routed.	I_T nexus through with the status was returned.
10b	All logical unit(s) contained within the SCSI device that contains the logical unit addressed by the command associated with the status.	All I_T nexus(es).
11b	Reserved	

The QUALIFIER field (see table 43) indicates additional information about the reason for the status code.

Table 43 — QUALIFIER field

Status code	QUALIFIER field	Description
All	0000h	No additional information (i.e., the same as returning no status qualifier).
BUSY	0001h - 3FEFh	The number of 100 milliseconds increments the application client should wait before sending another command to the logical unit(s) indicated by the SCOPE field using the nexus(es) indicated by the SCOPE field.
	3FF0h - 3FFDh	Reserved
	3FFEh	The application client should stop sending commands to the logical unit(s) indicated by the SCOPE field using the nexus(es) indicated by the SCOPE field.
	3FFFh	The logical unit(s) indicated by the SCOPE field are not able to accept the command because they are servicing too many other I_T nexuses.
TASK SET FULL [a]	0001h - 3FEFh	The application client should wait before sending another command to the logical unit on any I_T nexus until: a) at least the number of 100 milliseconds increments indicated in the QUALIFIER field have elapsed; or b) a command addressed to the logical unit on any I_T nexus completes or terminates.
	3FF0h - 3FFFh	Reserved
GOOD	0001h - 3FFFh	Reserved
CHECK CONDITION	0001h - 3FEFh	The number of 100 milliseconds increments the application client should wait before sending another command to the logical unit(s) indicated by the SCOPE field using the nexus(es) indicated by the SCOPE field.
	3FF0h - 3FFFh	Reserved
CONDITION MET	0001h - 3FFFh	Reserved
RESERVATION CONFLICT	0001h - 3FFFh	Reserved
ACA ACTIVE	0001h - 3FFFh	Reserved
TASK ABORTED	0001h - 3FFFh	Reserved
All others	0001h - 3FFFh	Reserved

[a] The SCOPE field shall be set to zero.

5.3.3 Status precedence

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

1) an ACA ACTIVE status;
2) a CHECK CONDITION status for any of the following unit attention conditions (i.e., with a sense key set to UNIT ATTENTION and one of the following additional sense codes):
   A) POWER ON, RESET, OR BUS DEVICE RESET OCCURRED;
   B) POWER ON OCCURRED;
   C) SCSI BUS RESET OCCURRED;
   D) MICROCODE HAS BEEN CHANGED;
   E) BUS DEVICE RESET FUNCTION OCCURRED;
   F) DEVICE INTERNAL RESET;
   G) COMMANDS CLEARED BY POWER LOSS NOTIFICATION; or
   H) I_T NEXUS LOSS OCCURRED;
3) a RESERVATION CONFLICT status;

   and

4) a status of:
   A) CHECK CONDITION, other than with a sense key set to ILLEGAL REQUEST or for any reason not listed in 2);
   B) GOOD;
   C) CONDITION MET; or
   D) TASK ABORTED.

NOTE 7 - The names of the unit attention conditions listed in this subclause (e.g., SCSI BUS RESET OCCURRED) are based on usage in SAM-2. The use of these unit attention condition names is not to be interpreted as a description of how the unit attention conditions are represented by any given SCSI transport protocol.

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

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

Any unit attention condition that was established for all logical units (e.g., REPORTED LUNS DATA HAS CHANGED) should be reported with a higher precedence than a CHECK CONDITION status with a sense key set to ILLEGAL REQUEST and an additional sense code set to LOGICAL UNIT NOT SUPPORTED.

5.4 SCSI transport protocol services for SCSI commands

5.4.1 SCSI transport protocol services for SCSI commands overview

The SCSI transport protocol services that support the **Execute Command** procedure call are described in 5.4. The following groups of SCSI transport protocol services are described:

a) the SCSI transport protocol services that support the delivery of the command and status (see 5.4.2); and
b) the SCSI transport protocol services that support the data transfers associated with processing a command (see 5.4.3).

5.4.2 Command and status SCSI transport protocol services

5.4.2.1 Command and status SCSI transport protocol services overview

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

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

5.4.2.2 Send SCSI Command SCSI transport protocol service request

An application client invokes the Send SCSI Command SCSI transport protocol service request to request that a SCSI initiator port send a command over the service delivery subsystem.

Send SCSI Command SCSI transport protocol service request:

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

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the command (see 4.8).

CDB: Command descriptor block (see 5.2).

Task Attribute: A value specifying one of the task attributes defined in 8.6. For specific requirements on the Task Attribute argument see 5.1.

Data-In Buffer Size: The number of bytes available for data transfers to the Data-In Buffer (see 5.4.3). SCSI transport protocols may interpret the Data-In Buffer Size to include both the size and the location of the Data-In Buffer.

Data-Out Buffer: A buffer containing command specific information to be sent to the logical unit (e.g., data or parameter lists needed to process the command (see 5.1)). The content of the Data-Out Buffer shall not change during the lifetime of the command (see 5.5) as viewed by the application client.

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

CRN: When CRN is used, all commands on an I_T_L nexus include a CRN argument (see 5.1).

Command Priority: The priority assigned to the command (see 8.7).

First Burst Enabled: An argument specifying that a SCSI transport protocol specific number of bytes from the Data-Out Buffer shall be delivered to the logical unit without waiting for the device server to invoke the **Receive Data-Out** SCSI transport protocol service.

5.4.2.3 SCSI Command Received SCSI transport protocol service indication

The task router (see 4.6.7) invokes the SCSI Command Received SCSI transport protocol service indication to notify a task manager that it has received a command over the service delivery subsystem.

SCSI Command Received SCSI transport protocol service indication:

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

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the command (see 4.8).

CDB: Command descriptor block (see 5.2).

Task Attribute: A value specifying one of the task attributes defined in 8.6. For specific requirements on the Task Attribute argument see 5.1.

CRN: When a CRN argument is used, all commands on an I_T_L nexus include a CRN argument (see 5.1).

Command Priority: The priority assigned to the command (see 8.7).

First Burst Enabled: An argument specifying that a SCSI transport protocol specific number of bytes from the Data-Out Buffer are being delivered to the logical unit without waiting for the device server to invoke the **Receive Data-Out** SCSI transport protocol service.

5.4.2.4 Send Command Complete SCSI transport protocol service response

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

Send Command Complete SCSI transport protocol service response**:**

Send Command Complete **(IN (I_T_L_Q Nexus, [Sense Data], [Sense Data Length], Status, Service Response, [Status Qualifier]))**

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the command (see 4.8).

Sense Data: If present, a Sense Data argument instructs the SCSI target port to complete with sense data to the SCSI initiator port (see 5.13).

Sense Data Length: The length in bytes of the sense data to be returned to the SCSI initiator port.

Status: Command completion status (see 5.1).

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

Status Qualifier: The Status Qualifier code for the command (see 5.3.2).

5.4.2.5 Command Complete Received SCSI transport protocol service confirmation

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

Command Complete Received SCSI transport protocol service confirmation over the service delivery subsystem:

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

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the command (see 4.8).

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

Sense Data: Sense data returned in the same I_T_L_Q nexus transaction as status (see 5.13).

Sense Data Length: The length in bytes of the received sense data.

Status: Command completion status (see 5.1).

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

Status Qualifier: The status qualifier for the command (see 5.3.2).

5.4.3 Data transfer SCSI transport protocol services

5.4.3.1 Introduction

The data transfer services described in 5.4.3 provide mechanisms for moving data to and from the SCSI initiator port while processing commands. 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 37). This standard allows either unidirectional or bidirectional data transfer. The processing of a command may require the transfer of data from the application client using the Data-Out Buffer, or to the application client using the Data-In Buffer, or both to and from the application client using both the Data-In Buffer and the Data-Out Buffer.

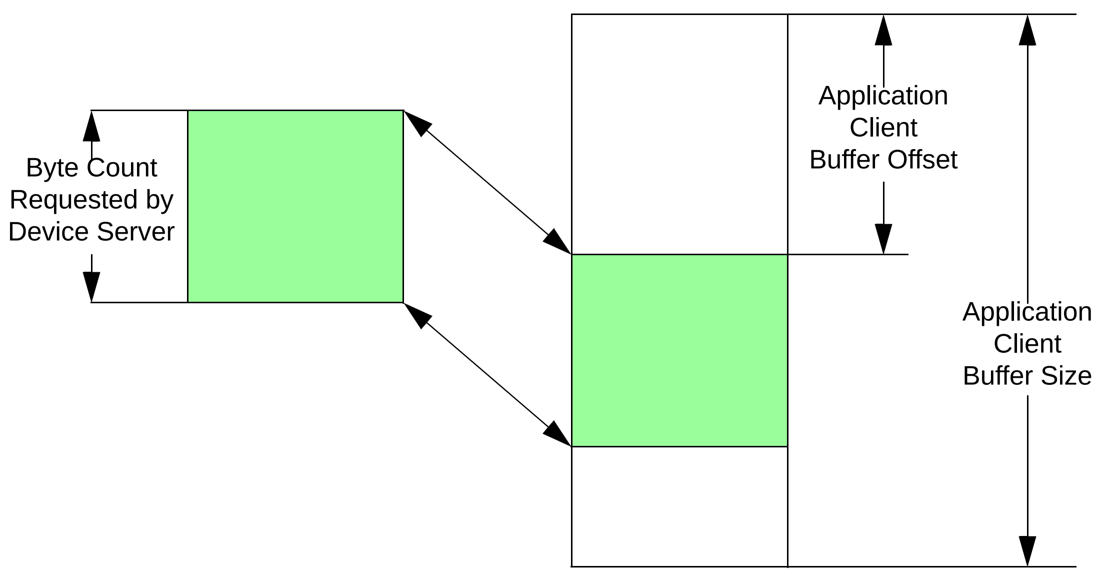


Figure 37 — Model for Data-In and Data-Out data transfers

This standard assumes that the buffering resources available to a logical unit are limited, and the buffer in the logical unit may not be capable of containing all of the data required to be transferred for one command. Such data needs to be moved between the application client and the logical unit in segments that are smaller than the transfer size specified in the command. The amount of data moved per segment is usually a function of the buffering resources available to the logical unit. Figure 37 shows the model for such incremental data transfers.

SCSI transport protocols may allow logical units to accept the initial portion of the Data-Out Buffer data, called the first burst, along with the command without waiting for the device server to invoke the **Receive Data-Out**

SCSI transport protocol service. This is modeled using **Receive Data-Out** protocol service calls for which the SCSI transport protocol may have moved the first burst prior to the call.

SCSI transport protocols that define a first burst capability shall include the First Burst Enabled argument in their definitions for the **Send SCSI Command** and **SCSI Command Received** SCSI transport protocol services. Logical units that implement the first burst capability shall implement the FIRST BURST SIZE field in the Disconnect-Reconnect mode page (see SPC-4).

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

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

Application Client Buffer Size: The total number of bytes in the application client's buffer (i.e., equivalent to Data-In Buffer Size for the Data-In Buffer or equivalent to Data-Out Buffer Size for the Data-Out Buffer).

Application Client Buffer Offset: Offset in bytes from the beginning of the application client's buffer (Data-In or Data-Out) to the first byte of transferred data.

Byte Count Requested by Device Server: Number of bytes to be moved by the data transfer request.

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

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

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

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

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

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

The STPL confirmed services specified in 5.4.3.4 are used by the task manager or device server to terminate partially completed transfers to the Data-In Buffer or from the Data-Out Buffer. The **Terminate Data Transfer** SCSI transport protocol service requests that one or more **Send Data-In** SCSI transport protocol service requests or **Receive Data-Out** SCSI transport protocol service requests be terminated by a SCSI target port.

5.4.3.2 Data-In delivery service

5.4.3.2.1 Send Data-In SCSI transport protocol service request

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

Send Data-In SCSI transport protocol service request:

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

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the command (see 4.8).

Device Server Buffer: The buffer in the device server from which data is to be transferred.

Application Client Buffer Offset: Offset in bytes from the beginning of the application client's buffer (i.e., the Data-In Buffer) to the first byte of transferred data.

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

5.4.3.2.2 Data-In Delivered SCSI transport protocol service confirmation

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

Data-In Delivered SCSI transport protocol service confirmation:

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

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

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the command (see 4.8).

Delivery Result: an encoded value representing one of the following:

DELIVERY SUCCESSFUL: The data was delivered successfully.

DELIVERY FAILURE: A service delivery subsystem error occurred while attempting to deliver the data.

5.4.3.3 Data-Out delivery service

5.4.3.3.1 Receive Data-Out SCSI transport protocol service request

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

Receive Data-Out SCSI transport protocol service request:

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

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the command (see 4.8).

Application Client Buffer Offset: Offset in bytes from the beginning of the application client's buffer (i.e., the Data-Out Buffer) to the first byte of transferred data.

Device Server Buffer: The buffer in the device server to which data is to be transferred.

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

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

5.4.3.3.2 Data-Out Received SCSI transport protocol service confirmation

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

Data-Out Received SCSI transport protocol service confirmation:

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

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

Input arguments:

I_T_L_Q Nexus: The I_T_L_Q nexus identifying the command (see 4.8).

Delivery Result: an encoded value representing one of the following:

DELIVERY SUCCESSFUL:The data was delivered successfully.

DELIVERY FAILURE: A service delivery subsystem error occurred while attempting to receive the data.

5.4.3.4 Terminate Data Transfer service

5.4.3.4.1 Terminate Data Transfer SCSI transport protocol service request

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

Terminate Data Transfer SCSI transport protocol service request:

Terminate Data Transfer (IN (Nexus))

Input argument:

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.8).

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

5.4.3.4.2 Data Transfer Terminated SCSI transport protocol service confirmation

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

Data Transfer Terminated SCSI transport protocol service confirmation:

Data Transfer Terminated (IN (Nexus))

Input argument:

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.8).

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

5.5 Command lifetime

This subclause specifies the events delimiting the beginning and end (i.e., lifetime) of a command from the viewpoint of the application client and device server. The task router and task manager have the same viewpoint of the beginning and end of a command as the device server.

An application client maintains an application client command from the time the **Send SCSI Command** SCSI transport protocol service request is invoked until:

a) the application client receives a service response of COMMAND COMPLETE for that command;
b) the application client receives notification of a unit attention condition with one of the following additional sense codes:
    A) any additional sense code whose ADDITIONAL SENSE CODE field is set to 2Fh (e.g., COMMANDS CLEARED BY ANOTHER INITIATOR, COMMANDS CLEARED BY POWER LOSS NOTIFICATION or COMMANDS CLEARED BY DEVICE SERVER), if in reference to the task set containing the command;
    B) any additional sense code whose ADDITIONAL SENSE CODE field is set to 29h (e.g., POWER ON, RESET, OR BUS DEVICE RESET OCCURRED; POWER ON OCCURRED; SCSI BUS RESET OCCURRED; BUS DEVICE RESET FUNCTION OCCURRED; DEVICE INTERNAL RESET; or I_T NEXUS LOSS OCCURRED); or
    C) MICROCODE HAS BEEN CHANGED;
c) the application client receives notification that the task manager has detected the use of a duplicate I_T_L_Q nexus (see 5.10);
d) the SCSI initiator device processes a SCSI device condition that aborts the command (see table 44);
e) the application client receives notification that the SCSI target device processed a SCSI device condition that aborted the command (see table 45);
f) a service response of FUNCTION COMPLETE for a task management function that aborted the command (see table 46);
g) a service response of COMMAND COMPLETE for another command that aborted the command (see table 47);
h) the application client receives a service response of FUNCTION COMPLETE following a QUERY TASK task management function directed to the specified command; or
i) the application client receives a service response of FUNCTION COMPLETE following a QUERY TASK SET task management function directed to the specified task set.

NOTE 8 - Items other than a) assume in-order delivery (see 4.4.3).

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

NOTE 9 - The names of the unit attention conditions listed in the subclause (e.g., SCSI BUS RESET OCCURRED) are based on usage in SAM-2. The use of these unit attention condition names is not to be interpreted as a description of how the unit attention conditions are represented by any given SCSI transport protocol.

The task manager creates a command upon receiving a **SCSI Command Received** SCSI transport protocol service indication (see 5.4.2.3) (i.e., upon processing the SCSI Command Received operation (see 4.6.20.2)).

The command shall exist until:

a) the device server sends a service response for the command of COMMAND COMPLETE; or

b) the command is aborted as described in 5.6.

When a SCSI transport protocol does not require state synchronization (see 4.4.2), 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 command as it appears to the application client is different from the lifetime observed by the device server.

Some commands initiate background operations that are processed after the command is no longer in the task set (i.e., status has been returned for the command) (e.g., a SEND DIAGNOSTIC command when used to initiate a background self-test (see SPC-4) or a write command when write cache is enabled (see SBC-3)). Background operations may be aborted by power on, hard reset, or logical unit reset. Unless otherwise specified (see 6.3.4 and 6.3.5), background operations shall not be aborted by I_T nexus loss or power loss expected.

Background operations may generate deferred errors that are reported in the sense data for a subsequent command (see SPC-4). Information that a deferred error occurred may be cleared before it is reported (e.g., by power on, hard reset, or logical unit reset). Deferred errors should not be cleared by I_T nexus loss or power loss expected.

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

5.6 Aborting commands

A command is aborted when a SCSI device condition (see 6.3), command, or task management function causes termination of the command prior to its completion by the device server. After a command is aborted and TASK ABORTED status, if any, is returned, the SCSI target device shall send no further requests or responses for that command.

See table 44 for a list of the SCSI device conditions that cause commands to be aborted in a SCSI initiator device.

Table 44 — SCSI device conditions that abort commands in a SCSI initiator device

SCSI device condition	Scope	Reference
Power on	All commands in the SCSI initiator device.	6.3.1
Hard reset	All commands with an I_T nexus involving the SCSI initiator port.	6.3.2
I_T nexus loss	All commands associated with the lost I_T nexus.	6.3.4
SCSI transport protocol specific conditions	As defined by the applicable SCSI transport protocol standard.	

See table 45 for a list of the SCSI device conditions that cause commands to be aborted in a SCSI target device.

Table 45 — SCSI device conditions that abort commands in a SCSI target device

SCSI device condition	Scope	Unit attention condition (see 5.14) additional sense code, if any	TASK ABORTED status [a]	Reference
Power on	All commands in the SCSI target device.	See table 54	No	6.3.1
Hard reset	All commands in all logical units to which the SCSI target port has access in the SCSI target device.	See table 54	Yes or no [c]	6.3.2
Logical unit reset [b]	All commands in the logical unit.	See table 54	Yes or no [d]	6.3.3 and 7.7
I_T nexus loss [b]	In each logical unit to which the SCSI target port has access, all commands associated with the lost I_T nexus.	See table 54	No	6.3.4 and 7.6
Power loss expected	All commands in the SCSI target device.	COMMANDS CLEARED BY POWER LOSS NOTIFICATION	No	6.3.5
SCSI transport protocol specific conditions	As defined by the applicable SCSI transport protocol standard.			

[a] “Yes” indicates that each command that is aborted on an I_T nexus other than the one that caused the SCSI device condition is completed with TASK ABORTED status, if the TAS bit is set to one in the Control mode page (see SPC-4). “No” indicates that no status is returned for aborted commands.
[b] This SCSI device condition is able to be invoked by a task management function listed in table 46.
[c] If the hard reset is caused by a particular I_T nexus (e.g., by a SCSI transport protocol-specific task management function), then “yes” applies. Otherwise, “no” applies.
[d] If the logical unit reset is caused by a particular I_T nexus (e.g., by a LOGICAL UNIT RESET task management function), then “yes” applies. Otherwise (e.g., if triggered by a hard reset), “no” applies.

See table 46 for a list of the task management functions that cause commands to be aborted.

Table 46 — Task management functions that abort commands

Task management function	Scope	Unit attention condition (see 5.14) additional sense code, if any [a]	TASK ABORTED status [b]	Reference
ABORT TASK (I_T_L_Q nexus)	Command specified by the I_T_L_Q Nexus argument.	None	No	7.2
ABORT TASK SET (I_T_L nexus)	All commands in the task set with the same I_T nexus as that specified by the I_T_L Nexus argument.	None	No	7.3
CLEAR TASK SET (I_T_L nexus)	All commands in the task set. [c]	COMMANDS CLEARED BY ANOTHER INITIATOR	Yes	7.5
LOGICAL UNIT RESET (I_T_L nexus)	See table 45 description of the logical unit reset condition.			
I_T NEXUS RESET (I_T nexus)	See table 45 description of the I_T nexus loss condition.			

[a] If the TAS bit is set to zero in the Control mode page (see SPC-4), the device server creates this unit attention condition for each I_T nexus that had command(s) aborted other than the I_T nexus that delivered the task management function. If the TAS bit is set to one in the Control mode page (see SPC-4), the device server does not create this unit attention condition.

[b] “Yes” indicates that each command that is aborted on an I_T nexus other than the one that delivered the task management function is completed with TASK ABORTED status, if the TAS bit is set to one in the Control mode page. “No” indicates that no status is returned for aborted commands.

[c] If the TST field is set to 001b (i.e., per I_T nexus) in the Control mode page (see SPC-4), then there is one task set per I_T nexus As a result, no other I_T nexuses are affected and CLEAR TASK SET is equivalent to ABORT TASK SET.

See table 47 for a list of the command related conditions that cause commands to be aborted.

Table 47 — Command related conditions that abort commands

Command related conditions	Scope	Unit attention condition (see 5.14) additional sense code, if any [a]	TASK ABORTED status [b]	Reference
CHECK CONDITION status if: a) the QERR field is set to 01b; and b) the TST field is set to 000b (i.e., shared) in the Control mode page (see SPC-4).	All commands in the task set.	COMMANDS CLEARED BY ANOTHER INITIATOR	Yes	5.8.3 and 5.9.2
CHECK CONDITION status if: a) the QERR field is set to 01b; and b) the TST field is set to 001b (i.e., per I_T nexus) in the Control mode page (see SPC-4).	All commands in the task set. [c]	None	No	5.8.3 and 5.9.2
Completion of a command with a CHECK CONDITION status if the QERR field is set to 11b in the Control mode page (see SPC-4).	All commands in the task set with the same I_T nexus as the command that was terminated.	None	No	5.8.3 and 5.9.2
Processing of a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action with a reservation key that is associated with the I_T nexus on which the command was received (see SPC-4).	All commands from all I_T nexuses with the specified reservation key.	COMMANDS CLEARED BY ANOTHER INITIATOR	Yes	SPC-4
Processing of a COPY OPERATION ABORT command that specifies a copy operation that is being processed in the foreground.	The copy operation originated by the third-party copy command specified by the LIST IDENTIFIER field in a COPY OPERATION ABORT command.	None	No	SPC-4
The return of an service response of SERVICE DELIVERY OR TARGET FAILURE.	The indicated command.	None	No	5.1
Termination of an overlapped command.	All commands with the same I_T nexus as the command that was terminated.	None	No	5.10

[a] If the TAS bit is set to zero in the Control mode page (see SPC-4), the device server creates this unit attention condition for each I_T nexus that had command(s) aborted other than the I_T nexus that delivered the task management function. If the TAS bit is set to one in the Control mode page (see SPC-4), the device server does not create this unit attention condition.

[b] “Yes” indicates that each command that is aborted on an I_T nexus other than the one that delivered the command is completed with TASK ABORTED status, if the TAS bit is set to one in the Control mode page (see SPC-4). “No” indicates that no status is returned for aborted commands.

[c] As a result of the TST field being set to 001b, there is one task set per I_T nexus, so no other I_T nexuses are affected.

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

When a device server receives a command or task management function on an I_T nexus that causes commands on the same I_T nexus to be aborted, the device server shall not return any notification that commands have been aborted other than:

a) a completion response for the command or task management function that caused the command(s) to be aborted; and
b) notification(s) associated with related effects of the command or task management function (e.g., a reset unit attention condition).

When a device server receives a command or task management function on an I_T nexus that causes commands on other I_T nexuses to be aborted, in addition to any notification (e.g., completion response) that a command or task management function generates on the I_T nexus on which command or task management function was received, the device server shall return any notifications for those commands on other I_T nexuses based on the setting of the TAS bit in the Control mode page (see SPC-4):

a) if the TAS bit is set to zero, the device server:
   A) shall not return status for the commands on other I_T nexuses that were aborted; and
   B) shall establish a unit attention condition for the SCSI initiator port associated with each I_T nexus containing commands on other I_T nexuses that were aborted with an additional sense code set as defined in table 46 and table 47;

   or

b) if the TAS bit is set to one, the device server:
   A) shall complete each aborted command on other I_T nexuses with a TASK ABORTED status; and
   B) shall not establish a unit attention condition for this reason.

When a logical unit completes one or more commands received on an I_T nexus with TASK ABORTED status, the logical unit should complete all of the affected commands before entering any other commands received on that I_T nexus into the task set.

5.7 Command processing example

A command is used to show the events associated with the processing of a single device service request (see figure 38). This example does not include error or exception conditions.

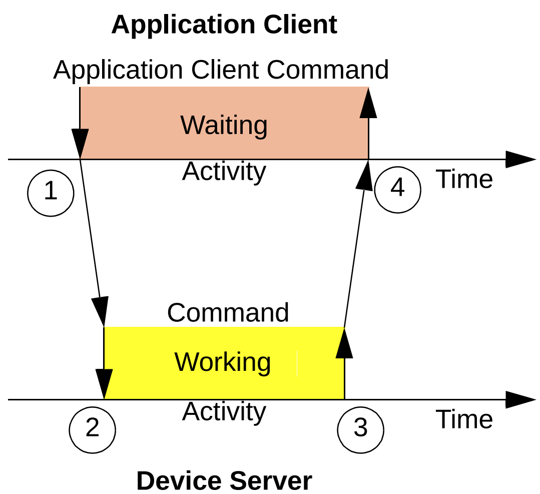


Figure 38 — Command processing events

The numbers in figure 38 identify the events described as follows:

1) the application client performs an **Execute Command** procedure call by invoking the **Send SCSI Command** SCSI transport protocol service request to send the CDB and other input parameters to the logical unit;
2) the task manager is notified through a **SCSI Command Received** SCSI transport protocol service indication containing the CDB and command parameters. A command 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) on command completion, the **Send Command Complete** SCSI transport protocol service response is invoked to return GOOD status and a service response of COMMAND COMPLETE; and
4) a **Command Complete Received** SCSI transport protocol service confirmation is passed to the application client by the SCSI initiator port.

5.8 Commands that complete with CHECK CONDITION status

5.8.1 Overview

An application client uses the NACA bit in the CONTROL byte of the CDB (see 5.2) to specify whether or not the device server establishes an ACA condition when it terminates a command with CHECK CONDITION status. The meaning of the value in the NACA bit is as follows:

a) If the NACA bit is set to zero, an ACA condition shall not be established; or
b) If the NACA bit is set to one, an ACA condition shall be established (see 5.9).

The requirements that apply when the ACA condition is not in effect are described in 5.8.2.

When a command terminates with a CHECK CONDITION status and an ACA condition is not established, commands other than the command completing with a the CHECK CONDITION status may be aborted as described in 5.8.3.

5.8.2 Handling commands when ACA is not in effect

Table 48 describes the handling of commands when an ACA condition is not in effect for the task set. The I_T nexuses that are associated with a task set are specified by the TST field in the Control mode page (see SPC-4).

Table 48 — Command handling when ACA is not in effect

New command properties		Device server action	ACA established if new command terminates with a CHECK CONDITION status
Task attribute [a]	NACA value [b]		
Any task attribute except the ACA task attribute	0	Process the command. [c]	No
	1		Yes
ACA task attribute	0	Process an invalid task attribute condition as described in 5.12.	No
	1		Yes

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

5.8.3 Aborting commands terminated with a CHECK CONDITION status without establishing an ACA

When a command terminates with a CHECK CONDITION status where the NACA bit is set to zero in the command's CDB CONTROL byte (i.e., when an ACA condition is not established), commands in the dormant command state or enabled command state (see 8.5) may be aborted based on the contents of the TST field and QERR field in the Control mode page (see SPC-4) as shown in table 49. The TST field specifies the type of task set in the logical unit. The QERR field specifies how the device server handles commands in the blocked command state and dormant command state when another command terminates with a CHECK CONDITION status.

Table 49 — Aborting commands when an ACA is not established

QERR	TST	Action
00b	000b	Commands other than the command terminated with a CHECK CONDITION status shall not be aborted.
	001b	
01b	000b	All enabled and dormant commands received on all I_T nexuses shall be aborted (see 5.6).
	001b	All enabled and dormant commands received on the I_T nexus on which the CHECK CONDITION status was returned shall be aborted (see 5.6). All commands received on other I_T nexuses shall not be aborted.
11b	000b	All enabled and dormant commands received on the I_T nexus on which the CHECK CONDITION status was returned shall be aborted (see 5.6). All commands received on other I_T nexuses shall not be aborted.
	001b	

5.9 Auto contingent allegiance (ACA)

5.9.1 ACA overview

The application client may request that the device server alter command processing when a command terminates with a CHECK CONDITION status by establishing an ACA condition using the NACA bit in the CONTROL byte (see 5.8.1).

The steps taken by the device server to establish an ACA condition are described in 5.9.2. Upon establishment of the ACA condition, some commands other than the command completing with the CHECK CONDITION status may be aborted and continued processing of other commands may be blocked as described in 5.9.2.

While the ACA condition is in effect and the TMF_ONLY bit is set to zero in the Control mode page (see SPC-4), new commands received by the logical unit from the faulted I_T nexus are not allowed to enter the task set unless they have the ACA task attribute (see 8.6.5). One of the results of the ACA task attribute requirement is that commands in-flight when the CHECK CONDITION status occurs are completed unprocessed with an ACA ACTIVE status. Multiple commands may be sent one at a time using the ACA task attribute to recover from the event that resulted in the ACA condition without clearing the ACA condition.

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

While the ACA condition is in effect:

a) new commands received on the faulted I_T nexus shall be handled as described in 5.9.3, and
b) new commands received on I_T nexuses other than the faulted I_T nexus shall be handled as described in 5.9.4.

The methods for clearing an ACA condition are described in 5.9.5.

5.9.2 Establishing an ACA

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

When an ACA condition is established, commands in the dormant command state or enabled command state (see 8.5) shall either be aborted or blocked based on the contents of the TST field and QERR field in the Control mode page (see SPC-4) as shown in table 50. The TST field specifies the type of task set in the logical unit. The QERR field specifies how the device server handles commands in the blocked command state and the dormant command state when another command terminates with a CHECK CONDITION status.

Table 50 — Blocking and aborting commands when an ACA is established

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

An ACA condition shall not cross task set boundaries and shall be preserved until it is cleared as described in 5.9.5.

If the SCSI transport protocol does not enforce state synchronization as described in 4.4.2, there may be a time delay between the occurrence of the ACA condition and the time at which the application client becomes aware of the condition.

5.9.3 Handling new commands received on the faulted I_T nexus when ACA is in effect

Table 51 describes the handling of new commands received on the faulted I_T nexus when ACA is in effect.

Table 51 — Handling for new commands received on a faulted I_T nexus during ACA

New command properties: Task attribute [a]	New command properties: NACA value [b]	ACA command present in the task set	TMF_ONLY value [c]	Device server action	ACA established if new command terminates with a CHECK CONDITION status
ACA task attribute	0	No	0	Process the command. [e]	No [d]
	1	No	0		Yes [d]
	n/a	n/a	1	Complete the command with ACA ACTIVE status.	n/a
	0 or 1	Yes	n/a		n/a
Any task attribute except the ACA task attribute	0 or 1	n/a	n/a	Complete the command with ACA ACTIVE status.	n/a

[a] Task attributes are described in 8.6.
[b] The NACA bit is in the CONTROL byte in the CDB (see 5.2).
[c] The TMF_ONLY bit is in the Control mode page (see SPC-4).
[d] If a command with the ACA task attribute terminates with a CHECK CONDITION status, the existing ACA condition shall be cleared and the value of the NACA bit shall control the establishment of a new ACA condition.
[e] All the conditions that affect the processing of commands (e.g., reservations) apply.

5.9.4 Handling new commands received on non-faulted I_T nexuses when ACA is in effect

5.9.4.1 Command processing that is permitted for commands received on a non-faulted I_T nexuses during ACA

The device server shall process a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action (see SPC-4) while an ACA condition is established when the command is received on a non-faulted I_T nexus.

NOTE 10 - The processing of specific commands (e.g., PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action) received on a non-faulted I_T nexus while an ACA condition is in effect provides SCSI initiator ports not associated with the faulted I_T nexus the opportunity to recover from error conditions that the SCSI initiator port associated with the faulted I_T nexus is unable to recover from itself.

5.9.4.2 Handling new commands received on non-faulted I_T nexuses when ACA is in effect

The handling of commands received on I_T nexuses other than the faulted I_T nexus depends on the value in the TST field in the Control mode page (see SPC-4).

Table 52 describes the handling of new commands received on I_T nexuses other than the faulted I_T nexus when ACA is in effect.

Table 52 — Handling for new commands received on non-faulted I_T nexuses during ACA

TST field value in Control mode page	New command properties: Task attribute [a]	New command properties: NACA value [b]	New command permitted during ACA [c]	Device server action	ACA established if new command terminates with a CHECK CONDITION status
000b	ACA task attribute	n/a	n/a	Complete the command with ACA ACTIVE status.	n/a
	Any task attribute except the ACA task attribute	0	No	Complete the command with BUSY status.	n/a
		1	No	Complete the command with ACA ACTIVE status.	n/a
		0	Yes	Process the command.	No [d]
		1	Yes		Yes [d]
001b	ACA task attribute	0	n/a	Process an invalid task attribute condition as described in 5.12.	No
		1			Yes
	Any task attribute except the ACA task attribute	0 or 1	n/a	Process the command. [e]	See 5.8.2.

[a] Task attributes are described in 8.6.
[b] The NACA bit is in the CONTROL byte in the CDB (see 5.2).
[c] See 5.9.4.1.
[d] If a permitted command terminates with CHECK CONDITION status, the existing ACA condition shall be cleared and the value of the NACA bit shall control the establishment of a new ACA condition.
[e] When the TST field in the Control mode page contains 001b, commands received on a non-faulted I_T nexus shall be processed as if the ACA condition does not exist (see 5.8.2). In this case, the logical unit shall be capable of handling concurrent ACA conditions and sense data associated with each I_T nexus.

5.9.5 Clearing an ACA condition

An ACA condition shall only be cleared:

a) as a result of a hard reset (see 6.3.2), logical unit reset (see 6.3.3), or I_T nexus loss (see 6.3.4);
b) by a CLEAR ACA task management function (see 7.4) received on the faulted I_T nexus;
c) by a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action with the ACA task attribute received on the faulted I_T nexus that clears the commands received on the faulted I_T nexus (see SPC-4);
d) by a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action with a task attribute other than the ACA task attribute received on a non-faulted I_T nexus that clears the commands received on the faulted I_T nexus;
e) when a command with the ACA task attribute received on the faulted I_T nexus terminates with CHECK CONDITION status; or
f) when a PERSISTENT RESERVE OUT command with a PREEMPT AND ABORT service action terminates with CHECK CONDITION status.

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

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

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

5.10 Overlapped commands

An overlapped command occurs when a task manager or a task router detects the use of a duplicate I_T_L_Q nexus ~~(see 4.6.6)~~ in a command before that I_T_L_Q nexus completes its command lifetime (see 5.5). Each SCSI transport protocol standard shall specify whether or not a task manager or a task router is required to detect overlapped commands.

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

> NOTE 11 - 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 terminate the overlapped command with the residue of blocks remaining to be written or read at the time the second command was received).

5.11 Incorrect logical unit numbers for commands

The SCSI target device's processing of~~response to a command addressed to~~ an incorrect logical unit number (see 4.7.1) is described in this subclause.

In response to a REQUEST SENSE command, a REPORT LUNS command, or an INQUIRY command the SCSI target device shall respond as defined in SPC-4.

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

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

   or

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

5.12 Task attribute exception conditions

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

> NOTE 12 - The use of the INVALID MESSAGE ERROR additional sense code is based on its similar usage in SAM-2. The use of the INVALID MESSAGE ERROR additional sense code is not to be interpreted as a description of how the task attributes are represented by any given SCSI transport protocol.

Task attribute support should be reported with the Extended INQUIRY Data VPD page (see SPC-4).

5.13 Sense data

5.13.1 Command terminated sense data or polled sense data

Sense data shall be made available by the logical unit when a command terminates with CHECK CONDITION status or other conditions (e.g., the processing of a REQUEST SENSE command). The format, content, and conditions under which sense data shall be prepared by the logical unit are specified in this standard, SPC-4, the applicable command standard, and the applicable SCSI transport protocol standard.

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

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

When a command terminates with CHECK CONDITION status, sense data shall be returned in the same I_T_L_Q nexus transaction as the CHECK CONDITION status. After the sense data is returned, it shall be cleared except when it is associated with a unit attention condition and the UA_INTLCK_CTRL field in the Control mode page (see SPC-4) contains 10b or 11b.

Completion with sense data in the same I_T_L_Q nexus transaction as a CHECK CONDITION status shall not affect ACA (see 5.9) or the sense data associated with a unit attention condition when the UA_INTLCK_CTRL field contains 10b or 11b.

5.13.2 Command completed sense data

Sense data may be made available by the logical unit when a command completes with a status other than CHECK CONDITION (e.g., GOOD). The format, content, and conditions under which sense data shall be prepared by the logical unit are specified in this standard, SPC-4, the applicable command standard, and the applicable SCSI transport protocol standard.

When a command completes with a status other than CHECK CONDITION (e.g., GOOD), the D_SENSE bit in the Control Mode page (see SPC-4) is set to one (i.e., returning descriptor formatted sense data is enabled), and sense data is available, that sense data shall be returned in the same I_T_L_Q nexus transaction as the status. After the sense data is returned, the sense data may be cleared.

5.14 Unit attention condition

5.14.1 Unit attention conditions that are not coalesced

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

a) a power on (see 6.3.1), hard reset (see 6.3.2), logical unit reset (see 6.3.3), I_T nexus loss (see 6.3.4), or power loss expected (see 6.3.5) occurs;
b) Commands received on this I_T nexus have been cleared by a command or a task management function associated with another I_T nexus and the TAS bit was set to zero in the Control mode page associated with this I_T nexus (see 5.6);
c) the logical unit inventory has been changed (see 4.6.17.1); or
d) any other event requiring the attention of the SCSI initiator device.

Unit attention conditions are classified by precedence levels. Table 53 defines the unit attention condition precedence levels.

Table 53 — Unit attention condition precedence level

Unit attention condition additional sense code	Unit attention condition precedence
POWER ON, RESET, OR BUS DEVICE RESET OCCURRED	highest
POWER ON OCCURRED or DEVICE INTERNAL RESET	
SCSI BUS RESET OCCURRED or MICROCODE HAS BEEN CHANGED or protocol specific	
BUS DEVICE RESET FUNCTION OCCURRED	
I_T NEXUS LOSS OCCURRED	
COMMANDS CLEARED BY POWER LOSS NOTIFICATION	
all others	Lowest

For unit attention conditions with the lowest precedence level with a given ADDITIONAL SENSE CODE field value, the unit attention condition with the ADDITIONAL SENSE CODE QUALIFIER field set to 00h has higher precedence level than the unit attention conditions with the ADDITIONAL SENSE CODE QUALIFIER field set to values other than 00h (e.g., PARAMETERS CHANGED has precedence over MODE PARAMETERS CHANGED and LOG PARAMETERS CHANGED). A unit attention condition with the lowest precedence level has equal priority with all unit attention conditions with the lowest precedence level with different ADDITIONAL SENSE CODE field values.

NOTE 13 - The unit attention additional sense code specificity order defined in 6.2 determines which unit attention condition is allowed to be established when certain conditions occur. The unit attention condition precedence defined in this subclause determines which unit attention conditions are allowed to clear other unit attention conditions if they have not yet been reported.

The device server shall maintain a queue of unit attention conditions of unspecified order for each I_T nexus. The queue should be large enough to hold every unit attention condition that the device server is capable of reporting.

When a device server establishes a unit attention condition:

1) the device server may clear unit attention conditions from the queue that are no longer needed as follows:
   A) the device server may clear any pending unit attention conditions in the queue that have lower precedence levels (e.g., BUS DEVICE RESET FUNCTION OCCURRED may clear I_T NEXUS LOSS OCCURRED and all unit attention conditions with a lower precedence); and
   B) the device server should clear pending unit attention conditions that have the same additional sense code (i.e., the device server should not add the same unit attention condition twice);
2) if a queue slot is available, then:
   A) if a higher precedence unit attention condition is not in the queue, the device server shall add the unit attention condition to the queue; or
   B) if a higher precedence unit attention condition is in the queue, the device server should add the unit attention condition to the queue.

   In the sense data for the unit attention condition, the device shall either:

   A) not include sense-key specific sense data; or
   B) include sense-key specific sense data and set the OVERFLOW bit to zero (see SPC-4);

   or

3) if a queue slot is not available, then the device server shall either:
   A) replace any unit attention condition in the queue; or

B) not add the unit attention condition to the queue.

The device server shall include sense-key specific sense data and set the OVERFLOW bit to one (see SPC-4) for at least one unit attention condition in the queue.

If the device server establishes multiple unit attention conditions as a result of the same event or a series of events, then it may establish the unit attention conditions in any order (e.g., in direct-access block devices, if a MODE SELECT command changes the initial command priority value, the device server may report PRIORITY CHANGED before MODE PARAMETERS CHANGED or may report MODE PARAMETERS CHANGED before PRIORITY CHANGED).

When the device server reports and clears a unit attention condition, it:

a) may select any unit attention condition in the queue to report; and
b) shall clear the unit attention condition from the queue after reporting it.

A unit attention condition shall persist until the device server clears the unit attention condition. Unit attention conditions are affected by the processing of commands as follows:

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

   If the UA_INTLCK_CTRL field in the Control mode page is set to 00b (see SPC-4), the SCSI target device shall clear any pending unit attention condition with an additional sense code of REPORTED LUNS DATA HAS CHANGED established for the SCSI initiator port associated with that I_T nexus in each logical unit accessible by the I_T nexus on which the REPORT LUNS command was received. Other pending unit attention conditions shall not be cleared.

   If the UA_INTLCK_CTRL field in the Control mode page contains 10b or 11b, the SCSI target device shall not clear any unit attention condition(s);

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

   If the device server has already generated the ACA condition (see 5.9) for a unit attention condition, the device server shall report the unit attention condition (i.e., option c)B) above);

d) if the device server supports the NOTIFY DATA TRANSFER DEVICE command (see ADC-2) and a NOTIFY DATA TRANSFER DEVICE command enters the enabled command state, then the device server shall process the NOTIFY DATA TRANSFER DEVICE command and shall neither report nor clear any unit attention condition; and
e) If a command other than INQUIRY, REPORT LUNS, REQUEST SENSE, or NOTIFY DATA TRANSFER DEVICE enters the enabled command state while a unit attention condition exists for the SCSI initiator port associated with the I_T nexus on which the command was received, the device server shall terminate the command with a CHECK CONDITION status. The device server shall provide sense data that reports a unit attention condition for the SCSI initiator port that sent the command on the I_T nexus.

   If a device server reports a unit attention condition with a CHECK CONDITION status and the UA_INTLCK_CTRL field in the Control mode page contains 00b (see SPC-4), then the device server shall clear the reported unit attention condition for the SCSI initiator port associated with that I_T nexus on the logical unit. If the unit attention condition has an additional sense code of REPORTED LUNS DATA HAS CHANGED, the SCSI target device shall clear any pending unit attention conditions with an additional sense code of REPORTED LUNS DATA HAS CHANGED established for the I_T nexus on which the command was received in each logical unit accessible by that I_T nexus. If the UA_INTLCK_CTRL field contains 10b or 11b, the device server shall not clear unit attention conditions reported with a CHECK CONDITION status.

5.14.2 Coalescing unit attention conditions

Within a logical unit conglomerate (see 4.6.10), the unit attention conditions associated with the following additional sense codes that occur in a subsidiary logical unit (see 4.6.12) shall be coalesced by the administrative logical unit (see 4.6.11) as described in this subclause:

a) ASYMMETRIC ACCESS STATE CHANGED; and
b) IMPLICIT ASYMMETRIC ACCESS STATE TRANSITION FAILED.

If a device server in a subsidiary logical unit is required to establish a unit attention that is reported using one of the additional sense codes listed in this subclause, then the device server shall:

a) not establish the unit attention condition; and
b) indicate to the administrative logical unit for the logical unit conglomerate that the administrative logical unit's device server is required to establish the unit attention condition.

If an administrative logical unit's device server is notified that it is required to establish a unit attention condition on behalf of one or more of the subsidiary logical units in the logical unit conglomerate, then the device server shall:

1) establish a unit attention condition for the additional sense code specified by the subsidiary logical unit or units and maintain it in a coalesced unit attentions queue that has all the properties of the queue described in 5.14.1 with the additional requirement that coalesced unit attention conditions have a separate identifying characteristic;
2) invoke the Reroute Conglomerate Task Management Functions operation (see 4.6.7.5) to reroute all task management functions to the administrative logical unit's task manager for processing as described in 4.6.11.2.4;
3) use the algorithm described in 4.6.11 to reroute commands associated with subsidiary logical units in the logical unit conglomerate in a way that causes the CHECK CONDITION status for the unit attention condition to be returned for a command sent to:
   A) the administrative logical unit; or
   B) any of the subsidiary logical units;

   and
4) after the unit attention condition has been returned, invoke the Stop Conglomerate Task Management Functions Rerouting operation (see 4.6.7.6) to end rerouting of task management functions within the logical unit conglomerate.

6 SCSI events and event notification model

6.1 SCSI events overview

SCSI events may occur or be detected in either:

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

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

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

When a SCSI port detects an event, it shall use the event notification services (see 6.4) to notify device servers, task managers, or application clients that the event has been detected.

The events detected and event notification services usage depends on whether the SCSI device is a SCSI target device (see figure 39) or a SCSI initiator device (see figure 40).

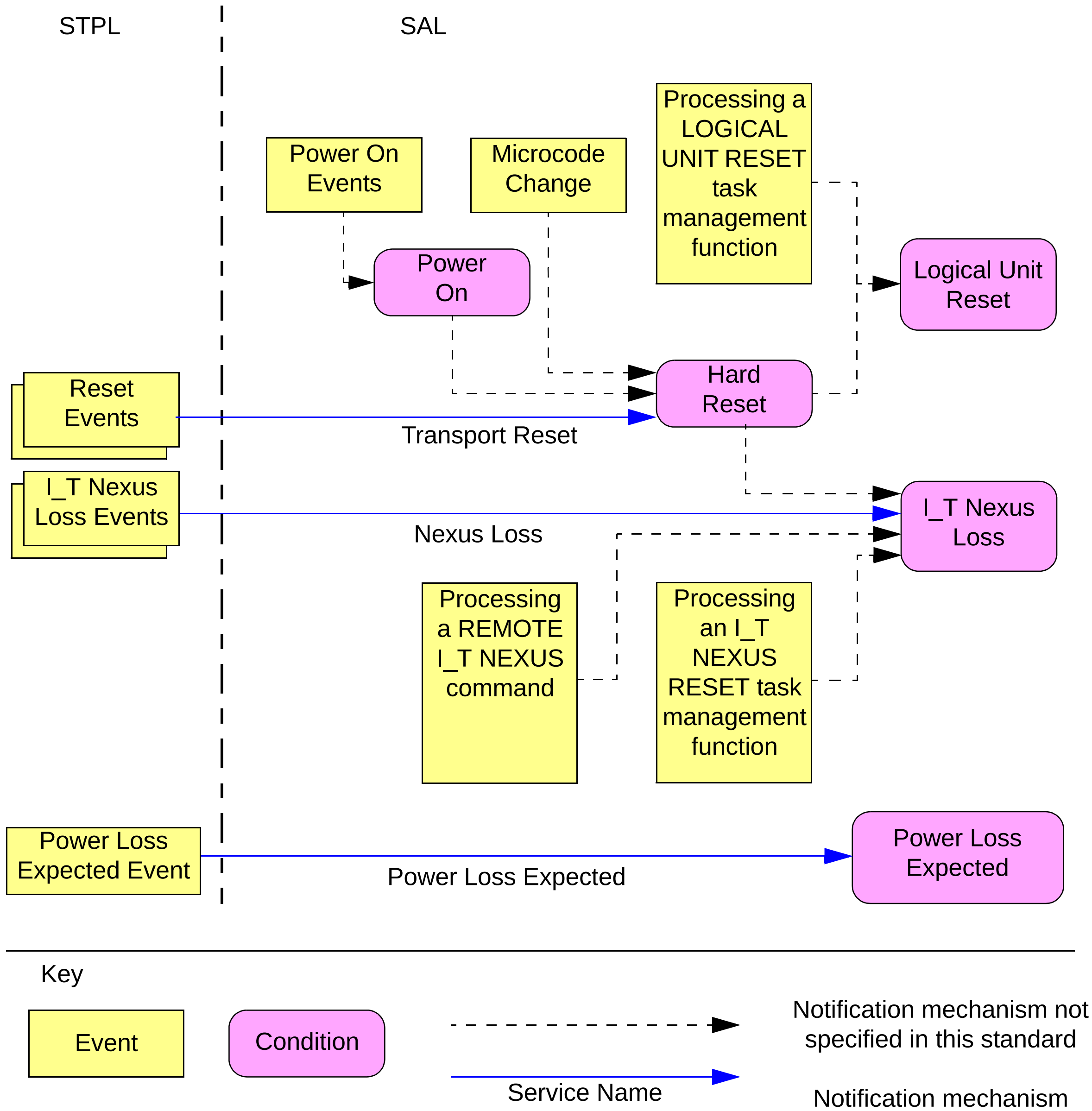


Figure 39 — Events and event notifications for SCSI target devices

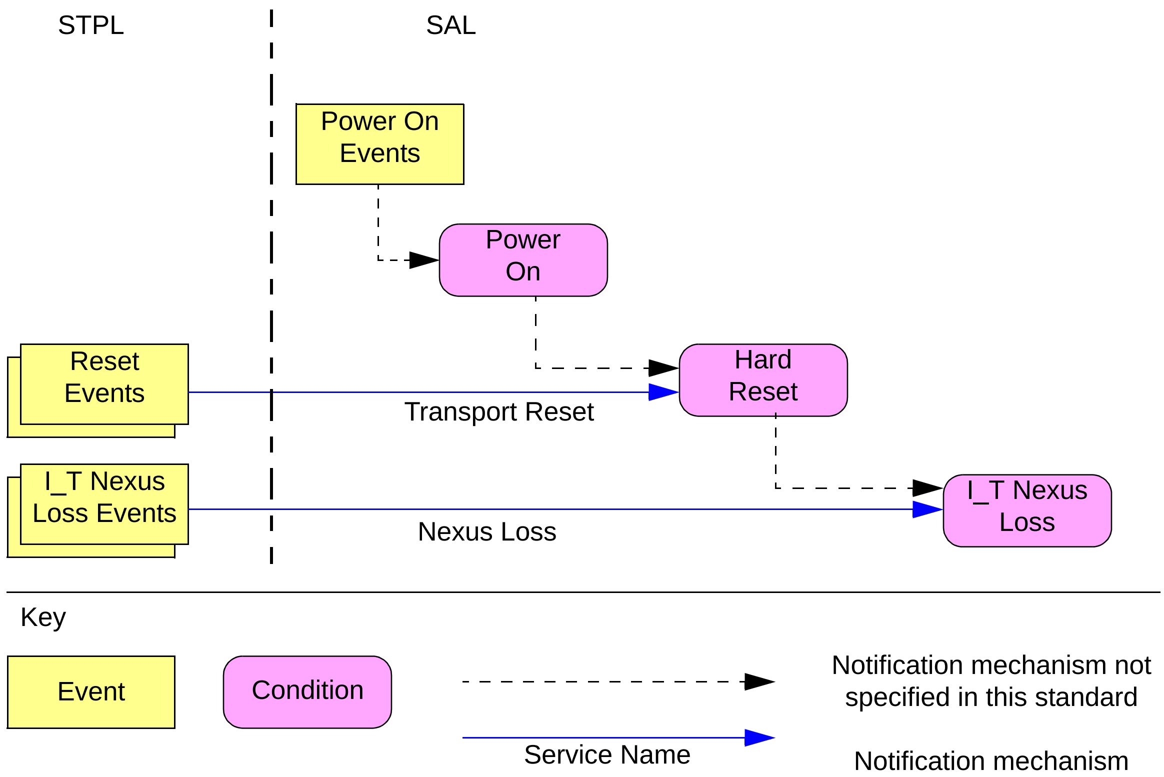


Figure 40 — Events and event notifications for SCSI initiator devices

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

Table 54 shows the additional sense code that a logical unit shall use when a unit attention condition (see 5.14) is established for each of the conditions shown in figure 39 (see 6.1). A SCSI transport protocol may define a more specific additional sense code than SCSI BUS RESET OCCURRED for reset events. The most specific condition in table 54 known to the logical unit should be used to establish the additional sense code for a unit attention.

The unit attention additional sense code specificity order defined in this subclause determines which unit attention condition is allowed to be established when certain conditions occur. The unit attention condition precedence defined in 5.14 determines which unit attention conditions are allowed to clear other unit attention conditions if they have not yet been reported.

Table 54 — Unit attention additional sense codes for events detected by SCSI target devices

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

[a] Used after a vendor-specific power on event has occurred (e.g., a firmware reboot).
[b] Only used if microcode has been changed (see SPC-4).
[c] Only used if a protocol-specific reset event has occurred.

NOTE 14 - The names of the unit attention conditions listed in the subclause (e.g., SCSI BUS RESET OCCURRED) are based on usage in SAM-2. The use of these unit attention condition names is not to be interpreted as a description of how the unit attention conditions are represented by any given SCSI transport protocol.

A logical unit should use the I_T NEXUS LOSS OCCURRED additional sense code when establishing a unit attention condition for an I_T nexus loss if:

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

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

6.3 Conditions resulting from SCSI events

6.3.1 Power on

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

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

Power on events include:

a) power being applied to the SCSI device; and
b) vendor-specific events that cause the SCSI device to behave as if power has been applied (e.g., firmware reboot).

6.3.2 Hard reset

Hard reset is a SCSI device condition resulting from:

a) a power on condition (see 6.3.1);
b) microcode change (see SPC-4); or

c) a reset event indicated by a **Transport Reset** event notification (see 6.4).

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

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

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

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

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

Although the task manager response to task management requests is subject to the presence of access restrictions, as managed by ACCESS CONTROL OUT commands (see SPC-4), a hard reset condition (see 6.3.2) shall not be prevented by access controls.

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

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

6.3.3 Logical unit reset

Logical unit reset is a logical unit condition resulting from:

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

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

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

a) abort all commands as described in 5.6;
b) abort all copy operations (see SPC-4);
c) terminate all task management functions;
d) clear all ACA conditions (see 5.9.5) in all task sets in the logical unit;
e) establish a unit attention condition (see 5.14 and 6.2);
f) initiate a logical unit reset for all dependent logical units (see 4.6.17.4); and
g) perform any additional functions required by the applicable command standards.

6.3.4 I_T nexus loss

An I_T nexus loss is a SCSI device condition resulting from:

a) a hard reset condition (see 6.3.2);
b) an I_T nexus loss event (e.g., logout) indicated by a **Nexus Loss** event notification (see 6.4);
c) indication that an I_T NEXUS RESET task management request (see 7.6) has been processed; or
d) an indication that a REMOVE I_T NEXUS command (see SPC-4) has been processed.

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

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

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

When a SCSI target port detects an I_T nexus loss, a **Nexus Loss** event notification shall be delivered to each logical unit to which the I_T nexus has access.

In response to an I_T nexus loss condition a logical unit shall take the following actions:

a) abort all commands received on the I_T nexus as described in 5.6;
b) abort all background third-party copy operations (see SPC-4);
c) terminate all task management functions received on the I_T nexus;
d) clear all ACA conditions (see 5.9.5) associated with the I_T nexus;
e) establish a unit attention condition for the SCSI initiator port associated with the I_T nexus (see 5.14 and 6.2); and
f) perform any additional functions required by the applicable command standards.

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

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

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

6.3.5 Power loss expected

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

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

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

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

a) the task manager in each logical unit to which the notification was delivered shall:
    A) abort all commands in all task sets (see 5.6), clearing all pending status and sense data;
    B) establish a unit attention condition as described in 5.6 for each initiator port associated with every I_T nexus on which a command was aborted (e.g., with the additional sense code set to COMMANDS CLEARED BY POWER LOSS NOTIFICATION); and
    C) abort all task management functions;
b) each logical unit to which the notification was delivered shall not change any other previously established conditions, including mode parameters, reservations, and ACA; and
c) the SCSI target device shall perform any additional functions required by the applicable SCSI transport protocol standards.

6.4 SCSI transport protocol services for event notification

6.4.1 SCSI transport protocol service for event notification overview

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

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

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

6.4.2 Nexus Loss SCSI transport protocol service indication

A SCSI target port invokes the **Nexus Loss** SCSI transport protocol service indication to notify a device server or task manager that an I_T nexus loss has occurred.

A SCSI initiator port invokes the **Nexus Loss** SCSI transport protocol service indication to notify an application client that an I_T nexus loss has occurred.

Nexus Loss (IN (I_T Nexus))

Input argument

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

6.4.3 Transport Reset SCSI transport protocol service indication

A SCSI target port invokes the Transport Reset SCSI transport protocol service indication to notify a device server or task manager that a hard reset has occurred.

A SCSI initiator port invokes the Transport Reset SCSI transport protocol service indication to notify an application client that a hard reset has occurred.

Transport Reset (IN (SCSI Port))

Input argument

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

6.4.4 Power Loss Expected SCSI transport protocol service indication

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

A SCSI target port invokes the Power Loss Expected SCSI transport protocol service indication to notify a device server or task manager that a power loss expected has occurred.

Power Loss Expected (IN (SCSI Port))

Input argument

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

7 Task management functions

7.1 Task management function procedure calls

An application client requests the processing of a task management function by invoking the SCSI transport protocol services described in 7.12, the collective operation of which is modeled in the following procedure call using the following format:

Service Response = Function name (IN (Nexus), OUT ([Additional Response Information])

The task management function names are summarized in table 55.

Table 55 — Task Management Functions

Task management function (i.e., function name)	Nexus argument	Additional Response Information argument supported	Reference
ABORT TASK	I_T_L_Q Nexus	no	7.2
ABORT TASK SET	I_T_L Nexus	no	7.3
CLEAR ACA	I_T_L Nexus	no	7.4
CLEAR TASK SET	I_T_L Nexus	no	7.5
I_T NEXUS RESET	I_T Nexus	no	7.6
LOGICAL UNIT RESET	I_T_L Nexus	no	7.7
QUERY TASK	I_T_L_Q Nexus	yes	7.8
QUERY TASK SET	I_T_L Nexus	no	7.9
QUERY ASYNCHRONOUS EVENT	I_T_L Nexus	yes	7.10

Input arguments:

Nexus: Contains an I_T Nexus argument, I_T_L Nexus argument, or I_T_L_Q Nexus argument (see 4.8) identifying the command or commands affected by the task management function.

I_T Nexus: The I_T nexus (see 4.8) affected by the task management function.

I_T_L Nexus: The I_T_L nexus (see 4.8) affected by the task management function.

I_T_L_Q Nexus: The I_T_L_Q nexus (see 4.8) affected by the task management function.

Output arguments:

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

One of the following SCSI transport protocol specific service responses shall be returned:

FUNCTION COMPLETE: A task manager response indicating that the requested function is complete. Unless another response is required, the task manager shall return this response upon completion of a task management request supported by the logical unit or SCSI target device to which the request was directed.

FUNCTION SUCCEEDED: A task manager response indicating that the requested function is supported and completed successfully. This task manager response shall only be used by functions that require notification of success (e.g., QUERY TASK, QUERY TASK SET, or QUERY ASYNCHRONOUS EVENT).

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

INCORRECT LOGICAL UNIT NUMBER: A task router response indicating that the function requested processing for an incorrect logical unit number (see 4.7.1).

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

Each SCSI transport protocol standard shall define the events for each of these service responses.

The task manager response to task management requests is subject to the presence of access restrictions, as managed by ACCESS CONTROL OUT and ACCESS CONTROL IN commands (see SPC-4), as follows:

a) a task management request of ABORT TASK, ABORT TASK SET, CLEAR ACA, I_T NEXUS RESET, QUERY TASK, QUERY TASK SET, or QUERY ASYNCHRONOUS EVENT shall not be affected by the presence of access restrictions;
b) a task management request of CLEAR TASK SET or LOGICAL UNIT RESET received from a SCSI initiator port that is denied access to the logical unit, either because it has no access rights or because it is in the pending-enrolled state, shall not cause any changes to the logical unit; and
c) the task management function service response shall not be affected by the presence of access restrictions.

7.2 ABORT TASK

Procedure call:

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

Description:

This function shall be supported by all logical units.

The task manager shall:

1) abort the specified command, if it exists, as described in (see 5.6); and
2) return a service response of FUNCTION COMPLETE.

The service response of FUNCTION COMPLETE does not distinguish between whether a command was aborted or no matching command was in the task set.

~~The task manager shall abort the specified command, if it exists, as described in 5.6.~~ Previously established conditions, including mode parameters, reservations, and ACA shall not be changed by the ABORT TASK task management function.

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

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

7.3 ABORT TASK SET

Procedure 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:

1) abort all commands in the task set that were received on the specified I_T nexus as described in 5.6; and
2) return a service response of FUNCTION COMPLETE.

Commands received on other I_T nexuses or in other task sets shall not be aborted. This task management function performed is equivalent to a series of ABORT TASK requests.

All pending status and sense data for the commands that were aborted shall be cleared. Other previously established conditions, including mode parameters, reservations, and ACA shall not be changed by the ABORT TASK SET function.

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

7.4 CLEAR ACA

Procedure 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 5.9).

For the CLEAR ACA task management function, the task set shall be the one defined by the TST field in the Control mode page (see SPC-4).

An application client requests a CLEAR ACA using the faulted I_T nexus (see 3.1.38) to clear an ACA condition (see 5.9) from the task set serviced by the logical unit. ~~The state of all commands in the task set shall be modified as described in 8.8. For a command with the ACA task attribute (see 8.6.5) receipt of a CLEAR ACA function shall have the same effect as receipt of an ABORT TASK function (see 7.2) specifying that command. If successful, this function shall be terminated with a service response of FUNCTION COMPLETE.~~

If there is an ACA condition and this function is received on the faulted I_T nexus, then the task manager shall:

1) abort each command in the task set with the ACA task attribute (see 8.6.5) as if an ABORT TASK function (see 7.2) specifying that command had occurred;
2) clear the ACA condition as described in 5.9.5; and
3) return a service response of FUNCTION COMPLETE.

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

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

If there is an ACA condition and this function is received on an I_T nexus other than the faulted I_T nexus, then the task manager shall return a service response of FUNCTION REJECTED.

~~If there is no ACA condition, then the service response to the CLEAR ACA task management function shall be FUNCTION COMPLETE.~~

If there is no ACA condition, then the task manager shall return a service response of FUNCTION COMPLETE.

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

7.5 CLEAR TASK SET

Procedure call:

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

Description:

This function shall be supported by all logical units.

The task manager shall:

1) abort all commands in the task set as described in 5.6; and
2) return a service response of FUNCTION COMPLETE.

If the TST field is set to 001b (i.e., per I_T nexus) in the Control mode page (see SPC-4), there is one task set per I_T nexus. As a result, no other I_T nexuses are affected and CLEAR TASK SET is equivalent to ABORT TASK SET (see 7.2).

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

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

7.6 I_T NEXUS RESET

Procedure call:

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

Description:

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

Each logical unit accessible through the SCSI target port shall perform the I_T nexus loss functions specified in 6.3.4 for the I_T nexus on which the function request was received, then:

1) the SCSI target device shall return a service response of FUNCTION COMPLETE ~~response~~; and
2) the logical unit(s) and the SCSI target port shall perform any additional functions specified by the SCSI transport protocol.

7.7 LOGICAL UNIT RESET

Procedure 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, t~~The logical unit shall:

1) perform the logical unit reset functions specified in 6.3.3; and
2) return a service response of FUNCTION COMPLETE.

NOTE 15 - Previous versions of this standard only required LOGICAL UNIT RESET support in logical units that supported hierarchical logical units.

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

7.8 QUERY TASK

Procedure call:

Service Response = QUERY TASK (IN (I_T_L_Q Nexus), OUT ([Additional Response Information]))

Description:

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

The task manager in the specified logical unit shall:

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

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

If the service response is FUNCTION SUCCEEDED, then the task manager shall set the Additional Response Information argument as defined in table 56.

Table 56 — Additional Response Information argument for QUERY TASK

Bit / Byte	7	6	5	4	3	2	1	0
0	(MSB)	PROGRESS INDICATION						
1								(LSB)
2	Reserved							

The PROGRESS INDICATION field is a percent complete indication of the specified command. The returned value is a numerator that has 65 536 (i.e., 10000h) as its denominator. The progress indication shall be based upon the total operation. A value of one indicates the minimum progress, and a value of FFFFh indicates the maximum progress. A value of zero indicates that the progress indicator is not supported

The progress indication should be time related. The granularity of these steps should be small enough to provide reasonable assurances to the application client that progress is being made.

7.9 QUERY TASK SET

Procedure call:

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

Description:

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

The task manager in the specified logical unit shall:

a) if there is any command present in the task set from the specified I_T nexus, then return a service response ~~set to~~of FUNCTION SUCCEEDED; or

b) if there is no command present in the task set from the specified I_T nexus, then return a service response ~~set to~~of FUNCTION COMPLETE.

7.10 QUERY ASYNCHRONOUS EVENT

Procedure call:

Service Response = QUERY ASYNCHRONOUS EVENT (IN (I_T_L Nexus), OUT ([Additional Response Information]))

Description:

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

The task manager in the specified logical unit shall:

a) if there is a unit attention condition (see 5.14) or a deferred error (see SPC-4) pending for the specified I_T nexus, then return a service response ~~set to~~of FUNCTION SUCCEEDED; or
b) if there is no unit attention condition or deferred error pending for the specified I_T nexus, then return a service response ~~set to~~of FUNCTION COMPLETE.

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

If the service response is FUNCTION SUCCEEDED, then the task manager shall set the Additional Response Information argument as defined in table 57.

Table 57 — Additional Response Information argument for QUERY ASYNCHRONOUS EVENT

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

The UADE DEPTH field indicates the number of pending unit attention conditions or deferred errors and is defined in table 58.

Table 58 — UADE DEPTH field

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

The SENSE KEY field indicates the value of the SENSE KEY field that would be returned in the sense data for the next unit attention condition or deferred error that is going to be reported (see SPC-4).

The ADDITIONAL SENSE CODE field indicates the value of the ADDITIONAL SENSE CODE field in the next unit attention condition or deferred error that is going to be reported (see SPC-4).

The ADDITIONAL SENSE CODE QUALIFIER field indicates the value of the ADDITIONAL SENSE CODE QUALIFIER field in the next unit attention condition or deferred error that is going to be reported (see SPC-4).

7.11 Task management function lifetime

The task manager creates a task management function upon receiving a **Task Management Request Received** SCSI transport service indication (see 7.12) (i.e., upon processing the Task Management Request Received operation (see 4.6.20.3)). The task management function shall exist until:

a) the task manager sends a service response for the task management function;
b) an I_T nexus loss (see 6.3.4);
c) a logical unit reset (see 6.3.3);
d) a hard reset (see 6.3.2);
e) power loss expected (see 6.3.5); or
f) a power on condition (see 6.3.1).

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

a) a service response of FUNCTION COMPLETE, FUNCTION SUCCEEDED, FUNCTION REJECTED, or INCORRECT LOGICAL UNIT NUMBER is received for that task management function;
b) notification of a unit attention condition with any additional sense code whose ADDITIONAL SENSE CODE field is set to 29h (e.g., POWER ON, RESET, OR BUS DEVICE RESET OCCURRED; POWER ON OCCURRED; SCSI BUS RESET OCCURRED; BUS DEVICE RESET FUNCTION OCCURRED; DEVICE INTERNAL RESET; or I_T NEXUS LOSS OCCURRED);
c) notification of a unit attention condition with an additional sense code set to MICROCODE HAS BEEN CHANGED; or
d) notification of a unit attention condition with an additional sense code set to COMMANDS CLEARED BY POWER LOSS NOTIFICATION.

NOTE 16 - Items other than a) assume in-order delivery (see 4.4.3).

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

NOTE 17 - The names of the unit attention conditions listed in the subclause (e.g., SCSI BUS RESET OCCURRED) are based on usage in SAM-2. The use of these unit attention condition names is not to be interpreted as a description of how the unit attention conditions are represented by any given SCSI transport protocol.

7.12 SCSI transport protocol services for task management functions

7.12.1 SCSI transport protocol services for task management functions overview

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

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.8).

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

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

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

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

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

7.12.2 Send Task Management Request SCSI transport protocol service request

An application client invokes the **Send Task Management Request** SCSI transport protocol service request to request that a SCSI initiator port send a task management function over the service delivery subsystem.

Send Task Management Request SCSI transport protocol service request:

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

Input arguments:

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.8).

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

7.12.3 Task Management Request Received SCSI transport protocol service indication

A task router (see 4.6.7) invokes the **Task Management Request Received** SCSI transport protocol service indication to notify a task manager that it has received a task management function over the service delivery subsystem.

Task Management Request Received SCSI transport protocol service indication:

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

Input arguments:

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.8).

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

7.12.4 Task Management Function Executed SCSI transport protocol service response

A task manager invokes the **Task Management Function Executed** SCSI transport protocol service response to request that a SCSI target port transmit task management function executed information over the service delivery subsystem.

Task Management Function Executed SCSI transport protocol service response:

Task Management Function Executed (IN (Nexus, Service Response, [Additional Response Information]))

Input arguments:

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.8).

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

FUNCTION COMPLETE:	The requested function has been completed.
FUNCTION SUCCEEDED:	The requested function is supported and completed successfully.
FUNCTION REJECTED:	The task manager does not implement the requested function.
INCORRECT LOGICAL UNIT NUMBER:	A task router response indicating that the function requested processing for an incorrect logical unit number (see 4.7.1).
SERVICE DELIVERY OR TARGET FAILURE:	The request was terminated due to a service delivery failure or SCSI target device malfunction. The task manager may or may not have successfully performed the specified function.

Additional Response Information: The Additional Response Information output argument for the task management function procedure call (see 7.1).

7.12.5 Received Task Management Function Executed SCSI transport protocol service confirmation

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

Received Task Management Function Executed SCSI transport protocol service confirmation:

Received Task Management Function Executed (IN (Nexus, Service Response, [Additional Response Information]))

Input arguments:

Nexus: An I_T nexus, I_T_L nexus, or I_T_L_Q nexus (see 4.8).

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

FUNCTION COMPLETE:	The requested function has been completed.
FUNCTION SUCCEEDED:	The requested function is supported and completed successfully.
FUNCTION REJECTED:	The task manager does not implement the requested function.
INCORRECT LOGICAL UNIT NUMBER:	A task router response indicating that the function requested processing for an incorrect logical unit number (see 4.7.1).
SERVICE DELIVERY OR TARGET FAILURE:	The request was terminated due to a service delivery failure or SCSI target device malfunction. The task manager may or may not have successfully performed the specified function.

Additional Response Information: The Additional Response Information output argument for the task management function procedure call (see 7.1).

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

7.13 Task management function example

Figure 41 shows the sequence of events associated with a task management function.

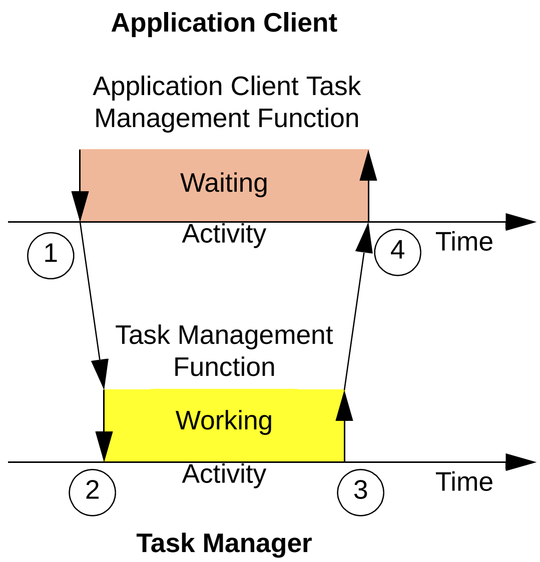


Figure 41 — Task management processing events

The numbers in figure 41 identify the events described as follows:

1) the application client 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** SCSI transport protocol service indication and begins processing the function;
3) the task manager performs the requested function and responds by invoking the **Task Management Function Executed** SCSI transport protocol service response to notify the application client of a service response of FUNCTION COMPLETE; and
4) a **Received Task Management Function Executed** SCSI transport protocol service confirmation is received by the application client.

8 Task set management

8.1 Introduction to task set management

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

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

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

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

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

a) a condition exists that causes that command to be completed with a status of BUSY, RESERVATION CONFLICT, TASK SET FULL, or ACA ACTIVE;
b) detection of an overlapped command (see 5.10) causes that command to be terminated with CHECK CONDITION status; or
c) SCSI transport protocol specific errors cause that command to be terminated with CHECK CONDITION status.

8.2 Implicit head of queue

A command standard (see 3.1.21) may define commands each of which may be processed by the task manager as if the command's task attribute is HEAD OF QUEUE task attribute even if the command is received with a SIMPLE task attribute or an ORDERED task attribute.

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

8.3 Command management model

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

a) the SIMPLE task attribute (see 8.6.1) shall be supported;
b) task attributes other than SIMPLE may be supported;
c) the QUEUE ALGORITHM MODIFIER field in the Control mode page (see SPC-4) shall control the processing sequence of commands having the SIMPLE task attribute;
d) the QERR field in the Control mode page (see SPC-4) shall control aborting of commands when any command terminates with CHECK CONDITION status; and
e) the CLEAR TASK SET task management function (see 7.5) shall be supported.

8.4 Command management events

Table 59 describes the events that cause changes in command state.

Table 59 — Command management events that cause changes in command state

Command management event	Description
All older commands completed:	If the TST field in the Control mode page (see SPC-4) equals 000b, all commands received on all I_T nexuses and accepted earlier in time than the referenced command have completed. If the TST field equals 001b, all commands received on the referenced I_T nexus and accepted earlier in time than the referenced command have completed.
All head of queue and older ordered commands completed:	If the TST field equals 000b, all the following commands received on all I_T nexuses have completed: a) all head of queue commands; and b) all ordered commands accepted earlier in time than the referenced command. If the TST field equals 001b, the following commands received on the referenced I_T nexus have completed: a) all head of queue commands; and b) all ordered commands accepted earlier in time than the referenced command.
ACA establishment:	An ACA condition has been established (see 5.8).
command abort:	A command has been aborted as described in 5.6.
command completion:	The device server has sent a service response of COMMAND COMPLETE for the command (see 5.1 and 5.5).
command completed:	A command has terminated or aborted.
ACA cleared:	An ACA condition has been cleared (see 5.9.5).

8.5 Command states

8.5.1 Overview

8.5.1.1 Command state nomenclature

This standard defines four command states, summarized in table 60.

Table 60 — Command state nomenclature

Command state name	Reference	Commands in this state may be called
Enabled command state	8.5.2	Enabled commands
Blocked command state	8.5.3	Blocked commands
Dormant command state	8.5.4	Dormant commands
Completed command state	8.5.5	Completed commands

8.5.1.2 Suspended information

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

8.5.2 Enabled command state

A command in the enabled command state may become a current command and may complete at any time, subject to the command completion constraints specified in the Control mode page (see SPC-4). A command that has been accepted into the task set shall not complete or become a current command unless it is in the enabled command state.

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

8.5.3 Blocked command state

A command in the blocked command state is prevented from completing due to an ACA condition. A command in this state shall not become a current command. While a command is in the blocked command state, any information the logical unit has or accepts for the command shall be suspended. If the TST field in the Control mode page (see SPC-4) equals 000b the blocked command state is independent of I_T nexus. If the TST field equals 001b the blocked command state applies only to the faulted I_T nexus.

8.5.4 Dormant command state

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

8.5.5 Completed command state

A command in the completed command state is removed from the task set.

8.5.6 Command states and command lifetimes

Figure 42 shows the events corresponding to two command processing sequences. Except for the dormant command state between times A and B in case 1, logical unit conditions and the commands processed by the command are identical. Assuming in each case the command completes with GOOD status at time C, the state observed by the application client for case 1 shall be indistinguishable from the state observed for case 2.

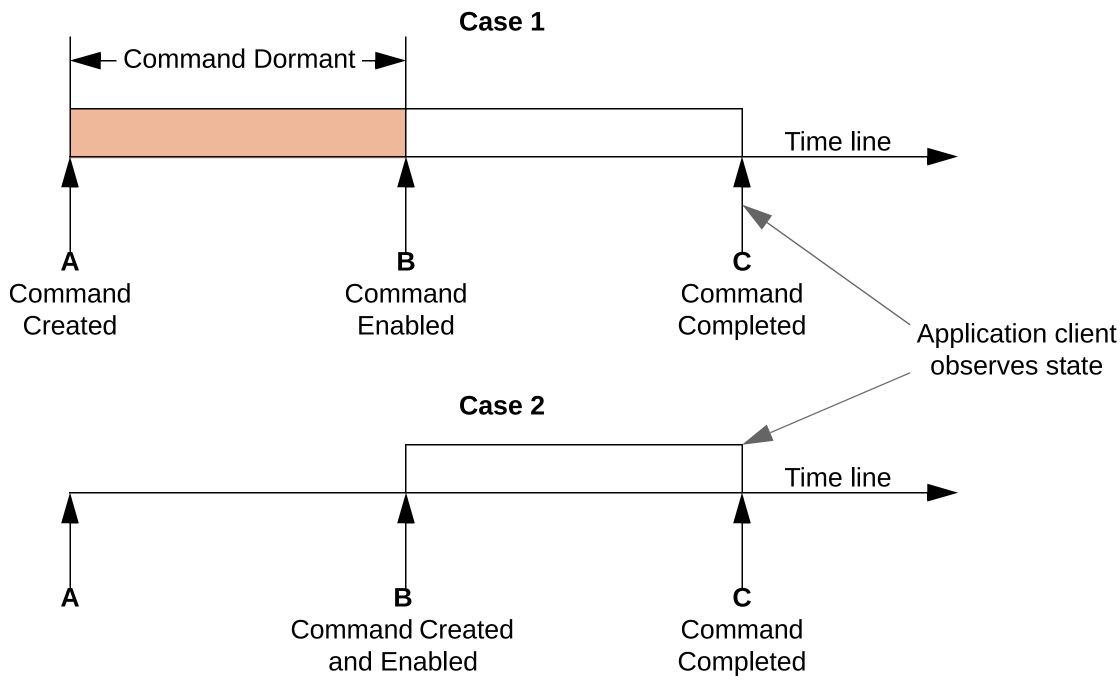


Figure 42 — Example of Dormant state command behavior

8.6 Task attributes

8.6.1 Overview

The application client shall assign a task attribute (see table 61) to each command.

Table 61 — Task attributes

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

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

8.6.2 Commands having the SIMPLE task attribute

If accepted, a command having the SIMPLE task attribute shall be entered into the task set in the dormant command state. The command shall not enter the enabled command state until all commands having a HEAD OF QUEUE task attribute and older commands having an ORDERED task attribute in the task set have completed (see 8.4).

The QUEUE ALGORITHM MODIFIER field in the Control mode page (see SPC-4) provides additional constraints on command completion order for commands having the SIMPLE task attribute.

8.6.3 Commands having the ORDERED task attribute

If accepted, a command having the ORDERED task attribute shall be entered into the task set in the dormant command state. The command shall not enter the enabled command state until all commands having a HEAD OF QUEUE task attribute and all older commands in the task set have completed (see 8.4).

8.6.4 Commands having the HEAD OF QUEUE task attribute

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

8.6.5 Commands having the ACA task attribute

If accepted, a command having the ACA task attribute shall be entered into the task set in the enabled command state. There shall be no more than one command having the ACA task attribute per task set (see 5.9.2).

8.7 Command priority

Command priority specifies the relative scheduling importance of a command having a SIMPLE task attribute in relation to other commands having SIMPLE task attributes already in the task set. If the command has a task attribute other than SIMPLE, then command priority is not used. Command priority is a value in the range of 0h through Fh. See table 62 for the scheduling importance of the command priority values.

Table 62 — Command priority

Value	Description
0h	A command with either no command priority or a command with a vendor-specific level of scheduling importance.
1h	A command with the highest scheduling importance.
•••	A command with decreased scheduling importance.
Fh	A command with the lowest scheduling importance.

If the Command Priority argument is set to zero or is not contained within the Send SCSI Received SCSI transport protocol service indication (see 5.4.2), and a priority has been assigned to the I_T_L nexus, then the device server shall use the specified priority for the I_T_L nexus as the command priority. A priority is assigned to an I_T_L nexus by a SET PRIORITY command (see SPC-4) or by the INITIAL COMMAND PRIORITY field in the Control Extension mode page (see SPC-4). If no priority has been assigned to the I_T_L nexus using the SET PRIORITY command and the logical unit does not support the INITIAL COMMAND PRIORITY field in the Control Extension mode page, then the device server shall set the command priority to 0h (i.e., vendor specific), or the command shall have no command priority.

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

8.8 Command state transitions

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

The command state diagram of figure 43 shows the behavior of a single command in response to an external event.

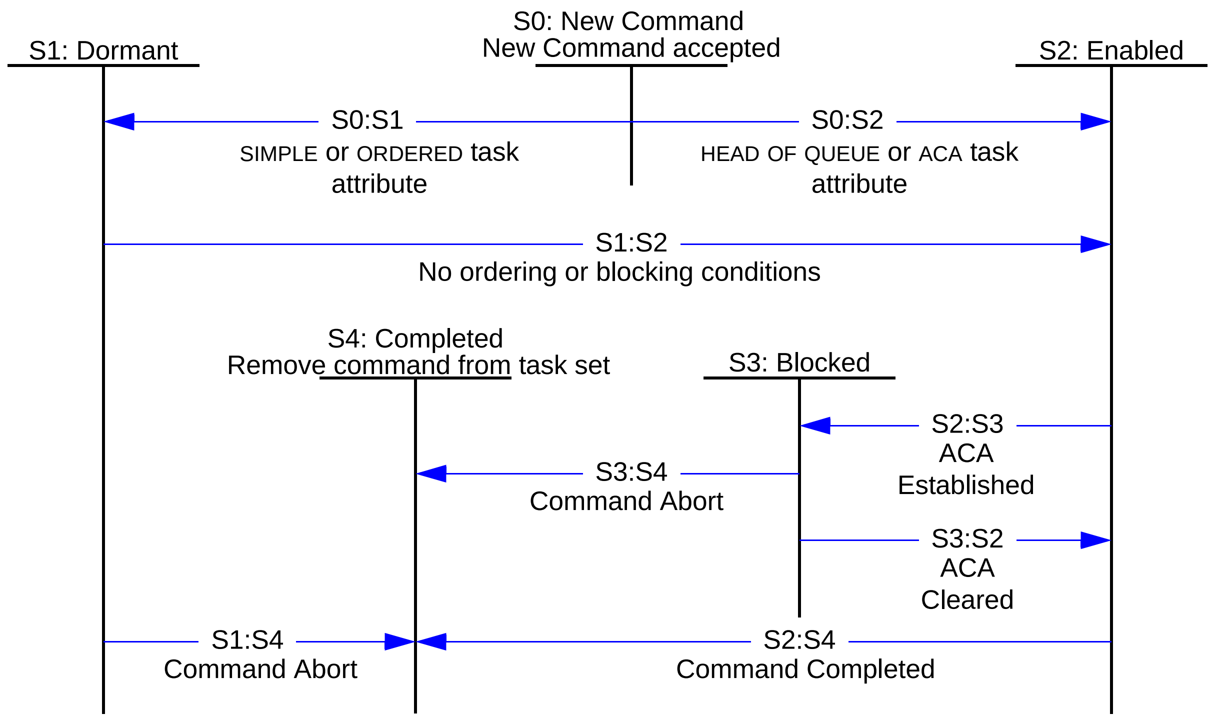


Figure 43 — Command states

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

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

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

a) a dormant command having the SIMPLE task attribute shall enter the enabled command state when all commands having a HEAD OF QUEUE task attribute and older commands having an ORDERED task attribute (see 8.4) have completed; or
b) a dormant command having the ORDERED task attribute shall enter the enabled command state when all commands having a HEAD OF QUEUE task attribute and all older commands (see 8.4) have completed.

If the TST field in the Control mode page (see SPC-4) contains 000b, then the transition from dormant command to enabled command shall not occur while an ACA is in effect for any I_T nexus (see 5.9.3 and 5.9.4). If the TST field contains 001b, then dormant commands from the faulted I_T nexus shall not transition to the enabled command state while an ACA is in effect for that I_T nexus (see 5.9.3).

Transition S2:S3: The establishment of an ACA condition (see 8.4) shall cause zero or more enabled commands to enter the blocked command state as described in 5.9.2.

Transition S3:S2: When an ACA condition is cleared (see 8.4), commands that entered the blocked command state when the ACA condition was established (see 5.9.2) shall re-enter the enabled command state.

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

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

8.9 Task set management examples

8.9.1 Introduction

Several task set management scenarios are shown in 8.9.2, 8.9.3, and 8.9.4. The examples are valid for configurations with one or more I_T nexuses~~multiple SCSI initiator ports~~ when the TST field contains 000b (i.e., the interaction among commands in a task set is independent of the I_T nexus on which a command is received). The examples are also valid for a single I_T nexus when the TST field contains 001b (i.e., task set management proceeds independently for each I_T nexus and the events and transitions for the task set associated with one I_T nexus do not affect the task set management for task sets associated with other I_T nexuses). Throughout these examples, the scope of the task set box drawn in each snapshot depends on the setting of the TST field in the Control mode page (see SPC-4).

The figure accompanying each example shows successive snapshots of a task set after various events (e.g., command creation or completion). In all cases, the constraints on command completion order established using Control mode page (see SPC-4) fields other that the TST field (e.g., the QUEUE ALGORITHM MODIFIER field) are not in effect.

A task set is shown as an ordered list or queue of commands with commands having a HEAD OF QUEUE task attribute towards the top of the figure. A new command having a HEAD OF QUEUE task attribute always enters the task set at the head, displacing older commands having a HEAD OF QUEUE task attribute. A command having a SIMPLE task attribute, ORDERED task attribute, or ACA task attribute always enters the task set at the end of the queue.

Command, denoted by rectangles, are numbered in ascending order from oldest to most recent. Fill, shape and line weight are used to distinguish command states and task attributes are shown in table 63.

Table 63 — Task attribute and state indications in examples

Task attribute	Box shape	Line weight	Command state
SIMPLE	Rounded Corners	Thin	Enabled
ORDERED	Square Corners	Thin	Dormant
HEAD OF QUEUE	Square Corners	Thick	Blocked
ACA	Square Corners	Thin Dashed	

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

8.9.2 Commands having the HEAD OF QUEUE task attribute

Figure 44 shows task set conditions when several commands having a HEAD OF QUEUE task attribute are processed.

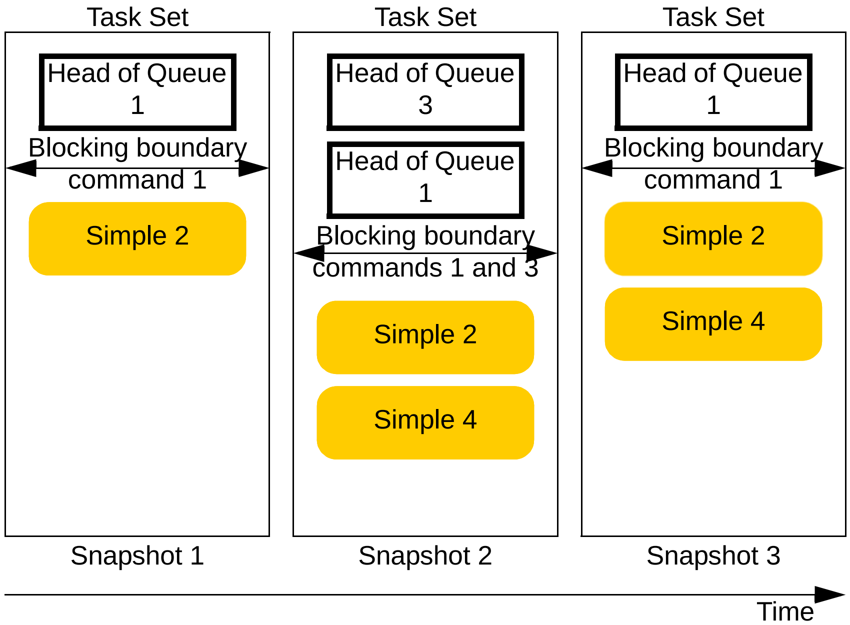


Figure 44 — Commands having the HEAD OF QUEUE task attribute and blocking boundaries (example 1)

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

Figure 45 is the same as the previous example, except that command 1 completes instead of command 3.

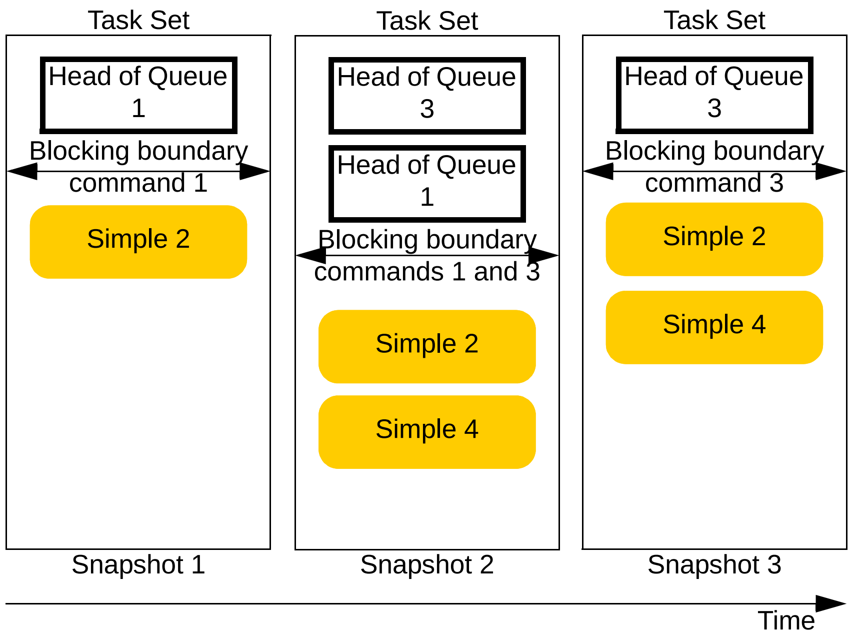


Figure 45 — Commands having the HEAD OF QUEUE task attribute and blocking boundaries (example 2)

Because the blocking boundary remains in place for a command having a HEAD OF QUEUE task attribute, both the commands having a SIMPLE task attribute remain in the dormant command state in snapshot 3. The blocking boundary is not removed until all commands having a HEAD OF QUEUE task attribute complete.

8.9.3 Commands having the ORDERED task attribute

An example of commands having ORDERED task attributes and commands having SIMPLE task attributes interaction is shown in figure 46.

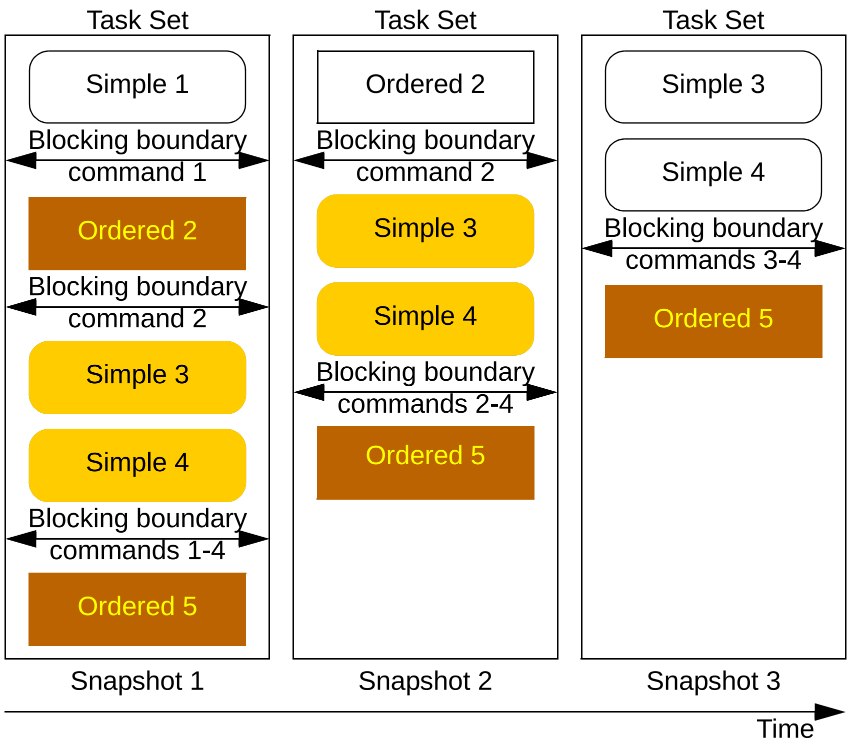


Figure 46 — Commands having ORDERED task attributes and blocking boundaries

The state of dormant command 2 through command 5 is determined by the requirements shown in table 64.

Table 64 — Dormant command blocking boundary requirements

<table>
<tr><th>Command</th><th>Reason for blocking boundary</th></tr>
<tr><td>2</td><td rowspan="2">A command having an ORDERED task attribute is not allowed to enter the enabled command state until all commands having a HEAD OF QUEUE task attribute and all older commands have completed.</td></tr>
<tr><td>5</td></tr>
<tr><td>3</td><td rowspan="2">A command having a SIMPLE task attribute is not allowed to enter the enabled command state until all commands having a HEAD OF QUEUE task attribute and all older ordered commands have completed.</td></tr>
<tr><td>4</td></tr>
</table>

The table 64 constraints are shown by the blocking boundaries in snapshot 1.

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

8.9.4 Commands having the ACA task attribute

Figure 47 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-4). Consequently, clearing an ACA condition does not cause commands to be aborted.

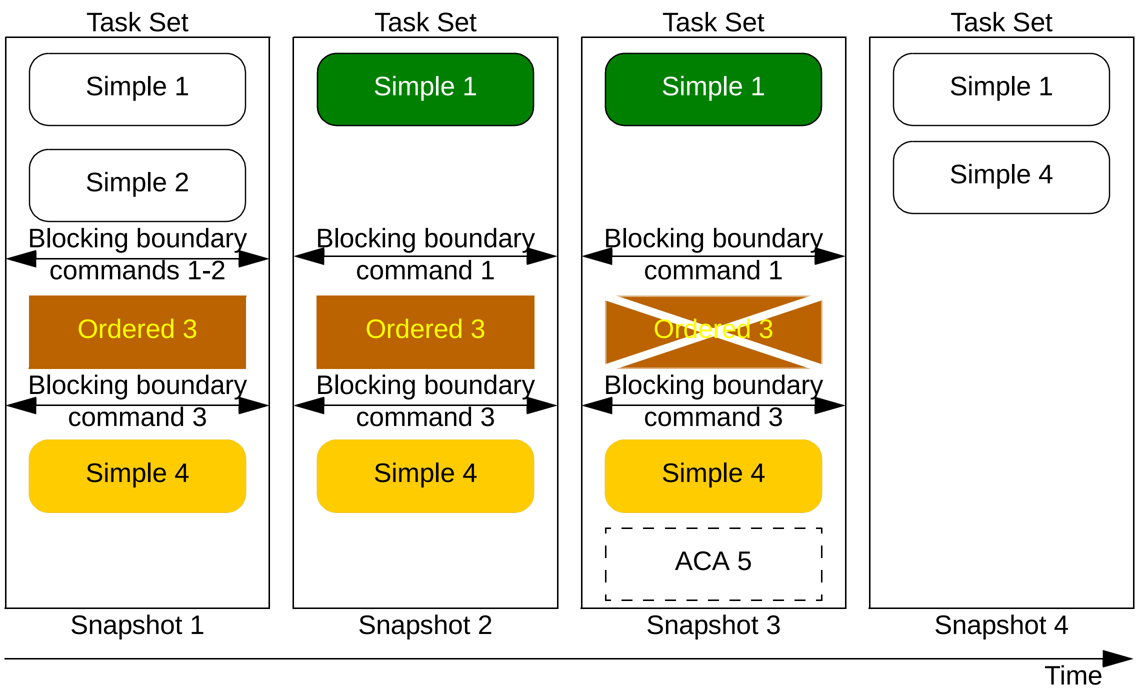


Figure 47 — Commands having ACA task attributes example

The completion of command 2 with CHECK CONDITION status causes command 1 to enter the blocked command state shown in snapshot 2. In snapshot 3, command having an ORDERED task attribute 3 is aborted using the ABORT TASK task management function and command having an ACA task attribute 5 is created to perform additional handling for the exception. Once the ACA condition is cleared (i.e., snapshot 4), command having a SIMPLE task attribute 1 is allowed to reenter the enabled command state. Since there are no commands having a HEAD OF QUEUE task attribute or older commands having an ORDERED task attribute, command 4 also transitions to the enabled command state.

Annex A
(informative)

Identifiers and names for objects

A.1 Identifiers and names overview

This annex summarizes identifiers and names.

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

a) SCSI initiator port;
b) SCSI target port;
c) logical unit; and
d) SCSI device.

A.2 Identifiers and names

This standard defines the identifier attributes and name attributes listed in A.1. The size requirements placed on identifier attributes by this standard are as shown in table A.1. This standard places no requirements on the sizes of name attributes as shown in table A.2.

Table A.1 — Identifier attribute size and support requirements

Attribute	Identifier	
	Size	Support requirements
Initiator port identifier	not specified	optional
Target port identifier	not specified	optional
LUN	2 bytes or 8 bytes [a]	mandatory

[a] Defined in the SCSI transport protocol.

Table A.2 — Name attribute size and support requirements

Attribute	Name	
	Size	Support requirements [a]
SCSI device name	not specified	optional
Initiator port name	not specified	optional
Target port name	not specified	optional
Logical unit name	not specified	mandatory (logical unit) none (well known logical unit)

[a] As defined in this standard.

Each SCSI transport protocol defines the size and format of identifier attributes and name attributes.

See table A.3 for a list of the sizes for identifier attributes for each SCSI transport protocol.

Table A.3 — Identifier attribute size for each SCSI transport protocol

Attribute	Size		
SCSI transport protocol	**Initiator port identifier**	**Target port identifier**	**LUN**
sADT (see ADT-2)	none	none	2 bytes
iADT (see ADT-2)	16 bytes	16 bytes	2 bytes
FCP (see FCP-4)	3 bytes	3 bytes	8 bytes
iSCSI (see iSCSI)	241 bytes [a]	233 bytes [a]	8 bytes
SBP (see SBP-3)	2 bytes	11 bytes	2 bytes
SOP (see SOP)	2 bytes	2 bytes	8 bytes
SAS SSP (see SPL-2)	8 bytes	8 bytes	8 bytes
SRP (see SRP)	16 bytes	16 bytes	8 bytes
UAS (see UAS)	none	15 bits	8 bytes

Key:
none = objects using this SCSI transport protocol do not have this attribute

[a] Maximum size, including the terminating null character byte.

See table A.4 for a list of the format of the identifier attributes for each SCSI transport protocol.

Table A.4 — Identifier attribute format for each SCSI transport protocol

Attribute	Format		
SCSI transport protocol	**Initiator port identifier**	**Target port identifier**	**LUN**
sADT (see ADT-2)	none	none	as specified in this standard (2 byte version only see 4.7)
iADT (see ADT-2)	IP address	IP address	as specified in this standard (2 byte version only see 4.7)
FCP (see FCP-4)	Fibre Channel address identifier	Fibre Channel address identifier	as specified in this standard (see 4.7)
iSCSI (see iSCSI)	iSCSI name [b] || ‘,i,’ hex prefix [c] || Initiator Session Identifier [d]	iSCSI name [b] || ‘,t,’ hex prefix [c] || Target Portal Group Tag [e]	as specified in this standard (see 4.7)
SBP (see SBP-3)	binary value	EUI-64 || Discovery ID [f]	as specified in this standard (2 byte version only see 4.7)
SOP (see SOP)	PCI Express routing ID	PCI Express routing ID	as specified in this standard (see 4.7)
SAS SSP (see SPL-2)	NAA IEEE Registered format or NAA Locally Assigned format	NAA IEEE Registered format or NAA Locally Assigned format	as specified in this standard (see 4.7)
SRP (see SRP)	EUI-64 || 8 byte extension [g]	EUI-64 || 8 byte extension [g]	as specified in this standard (see 4.7)
UAS (see UAS)	none	USB device address || USB interface number	as specified in this standard (see 4.7)

Key:
|| = “concatenated with”
none = objects using this SCSI transport protocol do not have this attribute

[a] iSCSI identifiers are concatenated strings containing no null characters except after the last string in the concatenation.
[b] The iSCSI name portion of the string is a worldwide unique UTF-8 string no more than 223 bytes long, not including null character termination.
[c] The hex prefix is a UTF-8 string containing ‘0x’ or ‘0X’.
[d] The Initiator Session Identifier (ISID) portion of the string is a UTF-8 encoded hexadecimal representation of a six byte binary value. This portion of the string contains no more than 12 bytes, not including null character termination if any, and contains Arabic numerals 0 through 9 and/or lower-case or upper-case English letters A through F.
[e] The Target Portal Group Tag (TPGT) portion of the string is a UTF-8 encoded hexadecimal representation of a two byte binary value. This portion of the string contains no more than 4 bytes, not including null character termination if any, and contains Arabic numerals 0 through 9 and/or lower-case or upper-case English letters A through F.
[f] See ISO/IEC 13213:1994 for more information on the Discovery ID.
[g] Required to be worldwide unique and recommend to be EUI-64 concatenated with an 8 byte extension.

See table A.5 for a list of the size of the name attributes for each SCSI transport protocol.

Table A.5 — Name attribute size for each SCSI transport protocol

Attribute	Size [a]			
SCSI transport protocol	**SCSI device name**	**Initiator port name**	**Target port name**	**Logical unit name**
sADT (see ADT-2)	not specified	none	none	as specified in the Device Identification VPD page (see SPC-4)
iADT (see ADT-2)	not specified	not specified	not specified	
FCP (see FCP-4)	8 bytes	8 bytes	8 bytes	
iSCSI [b] (see iSCSI)	224 bytes	241 bytes	233 bytes	
SBP (see SBP-3)	not specified	8 bytes	11 bytes	
SOP (see SOP)	8 bytes	8 bytes	8 bytes	
SAS SSP (see SPL-2)	8 bytes	none	none	
SRP (see SRP)	not specified	16 bytes	16 bytes	
UAS (see UAS)	not specified	none	none	8 bytes or 16 bytes [c]

Key:
not specified = objects using this SCSI transport protocol may or may not have this attribute and the SCSI transport protocol standard does not specify a size for this attribute
none = objects using this SCSI transport protocol do not have this attribute

[a] Any SCSI transport protocol may support the SCSI name string format (see SPC-4), resulting in names with the sizes shown in the iSCSI column.
[b] Maximum size, including the terminating null character byte.
[c] if the NAA is 2h (i.e., IEEE Extended), 3h (i.e., Locally Assigned), or 5h (i.e., IEEE Registered), then the size is 8 bytes; if the NAA is 6h (i.e., IEEE Registered Extended), then the size is 16 bytes

See table A.6 for a list of the format of the name attributes for each SCSI transport protocol.

Table A.6 — Name attribute format for each SCSI transport protocol

<table>
<tr><th>Attribute</th><th colspan="4">Format [a]</th></tr>
<tr><th>SCSI transport protocol</th><th>SCSI device name</th><th>Initiator port name</th><th>Target port name</th><th>Logical unit name</th></tr>
<tr><td>sADT
(see ADT-2)</td><td>not specified</td><td>none</td><td>none</td><td rowspan="8">as specified in the Device Identification VPD page name (see SPC-4)</td></tr>
<tr><td>iADT
(see ADT-2)</td><td>not specified</td><td>not specified</td><td>not specified</td></tr>
<tr><td>FCP
(see FCP-4)</td><td>NAA IEEE Registered format</td><td>Fibre Channel Name_Identifier</td><td>Fibre Channel Name_Identifier</td></tr>
<tr><td>iSCSI
(see iSCSI)</td><td>SCSI name string format</td><td>iSCSI name [c] || ‘,i,’ hex prefix [d] || Initiator Session Identifier [e]</td><td>iSCSI name [c] || ‘,t,’ hex prefix [d] || Target Portal Group Tag [f]</td></tr>
<tr><td>SBP
(see SBP-3)</td><td>not specified</td><td>EUI-64</td><td>EUI-64 || Discovery ID [g]</td></tr>
<tr><td>SOP
(see SOP)</td><td>NAA IEEE Registered format or NAA Locally Assigned format</td><td>NAA IEEE Registered format or NAA Locally Assigned format</td><td>NAA IEEE Registered format or NAA Locally Assigned format</td></tr>
<tr><td>SAS SSP
(see SPL-2)</td><td>NAA IEEE Registered format or NAA Locally Assigned format</td><td>none</td><td>none</td></tr>
<tr><td>SRP
(see SRP)</td><td>not specified</td><td>EUI-64 || 8 byte extension [h]</td><td>EUI-64 || 8 byte extension [h]</td></tr>
<tr><td>UAS
(see UAS)</td><td>not specified</td><td>none</td><td>none</td><td>NAA format</td></tr>
<tr><td colspan="5">Key:
|| = “concatenated with”
not specified = objects using this SCSI transport protocol may or may not have this attribute and the SCSI transport protocol standard does not specify a size for this attribute
none = objects using this SCSI transport protocol do not have this attribute</td></tr>
<tr><td colspan="5">[a] In addition to the name formats shown in this table, any SCSI transport protocol may support the SCSI name string format (see SPC-4).
[b] iSCSI identifiers are concatenated strings containing no null characters except after the last string in the concatenation.
[c] The iSCSI name portion of the string is a worldwide unique UTF-8 string no more than 223 bytes long, not including null character termination.
[d] The hex prefix is a UTF-8 string containing ‘0x’ or ‘0X’.
[e] The Initiator Session Identifier (ISID) portion of the string is a UTF-8 encoded hexadecimal representation of a six byte binary value. This portion of the string contains no more than 12 bytes, not including null character termination if any, and contains Arabic numerals 0 through 9 and/or lower-case or upper-case English letters A through F.
[f] The Target Portal Group Tag (TPGT) portion of the string is a UTF-8 encoded hexadecimal representation of a two byte binary value. This portion of the string contains no more than 4 bytes, not including null character termination if any, and contains Arabic numerals 0 through 9 and/or lower-case or upper-case English letters A through F.
[g] See ISO/IEC 13213:1994 for more information on the Discovery ID.
[h] Required to be worldwide unique and recommended to be EUI-64 concatenated with an 8 byte extension.</td></tr>
</table>

Annex B
(informative)

SCSI Initiator Port attributes and SCSI Target Port attributes supported by SCSI transport protocols

Table B.1 and table B.2 lists the values of the SCSI Initiator Port attributes that a SCSI initiator port using each different SCSI transport protocol is able to return, and the values of the SCSI Target Port attributes that a SCSI target port using that SCSI transport protocol is able to return.

Table B.1 — SCSI Initiator Port attributes and SCSI Target Port attributes that are supported by ADT-2, FCP-4, and iSCSI SCSI transport protocols

Attribute	ADT (see ADT-2)	FCP (see FCP-4)	iSCSI (see iSCSI)
LUN size (in bits)	16	64	64
Maximum CDB length [a] (in bytes)	65 523	268	65 550
Command identifier size (in bits)	3	16	32
Task Attributes supported	SIMPLE, HEAD OF QUEUE, ORDERED, and ACA		
Maximum Data-In Buffer Size (in bytes)	FFFF FFFFh		
Maximum Data-Out Buffer Size (in bytes)	FFFF FFFFh		
Maximum CRN [b]	zero	FFh	zero
Command Priority supported	no	yes	no
Maximum Sense Data Length [c] (in bytes)	FFFFh	FFFF FFFFh	FFFFh
Status Qualifier supported	no	yes	no
Additional Response Information supported	no	yes	no
Bidirectional Commands supported	yes		
Task Management Functions supported [d]	ABORT TASK, ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, CLEAR ACA, QUERY TASK	ABORT TASK, ABORT TASK SET, CLEAR TASK SET, CLEAR ACA, LOGICAL UNIT RESET, QUERY TASK, QUERY TASK SET, QUERY ASYNCHRONOUS EVENT	ABORT TASK, ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, CLEAR ACA

[a] SPC-4 defines the maximum length of a CDB as being 260 bytes.
[b] Maximum CRN of zero indicates that CRN is not supported.
[c] SPC-4 defines the maximum length of sense data as being 252 bytes.
[d] The task management function name is the name of the procedure call, not an argument to a procedure call.

Table B.2 — SCSI Initiator Port attributes and SCSI Target Port attributes that are supported by SOP, ~~SPL~~SAS SSP, SRP, and UAS SCSI transport protocols

Attribute	SOP (see SOP)	SAS SSP (SPL-2)	SRP (see SRP)	UAS (see UAS)
LUN size (in bits)	64	64	64	64
Maximum CDB length [a] (in bytes)	268	268	268	268
Command identifier size (in bits)	16	16	64	16
Task Attributes supported	SIMPLE, HEAD OF QUEUE, ORDERED, and ACA			
Maximum Data-In Buffer Size (in bytes)	FFFF FFFFh			none
Maximum Data-Out Buffer Size (in bytes)	FFFF FFFFh			none
Maximum CRN [b]	zero			
Command Priority supported	yes	yes	no	yes
Maximum Sense Data Length [c] (in bytes)	FFFFh	1 000	FFFF FFFFh	252
Status Qualifier supported	yes	yes	no	yes
Additional Response Information supported	yes	yes	no	yes
Bidirectional Commands supported	yes			
Task Management Functions supported [d]	all	all	ABORT TASK, ABORT TASK SET, CLEAR TASK SET, LOGICAL UNIT RESET, CLEAR ACA	all

[a] SPC-4 defines the maximum length of a CDB as being 260 bytes.
[b] Maximum CRN of zero indicates that CRN is not supported.
[c] SPC-4 defines the maximum length of sense data as being 252 bytes.
[d] The task management function name is the name of the procedure call, not an argument to a procedure call.

Annex C
(informative)

Terminology mapping to SAM-3

The introduction of a UML model into this standard resulted in changes in terminology between this standard and SAM-3 (see table C.1).

Table C.1 — Terminology mapping to SAM-3

Term used in this standard	Term used in SAM-3
command identifier	task tag
command	task

Annex D
(informative)

SCSI transport protocol acronyms

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.

NAA: Name Address Authority (see SPC-4).

SAS SSP: SAS Serial SCSI Protocol (see SPL-2).

UTF-8: See ISO/IEC 10646-1:2000.

Bibliography

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

ISO/IEC 14776-412, *Information technology - Small computer system interface (SCSI) - Part 412: Architecture model-2 (SAM-2)*.

ISO/IEC 14776-413, *Information technology - Small computer system interface (SCSI) - Part 412: Architecture model-3 (SAM-3)*.

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/*.

ISO/IEC 14776-262, *Information technology - Small computer system interface (SCSI) - Part 261: SAS Protocol Layer (SPL-2)*