
Information technology - SCSI / NVMe Translation (SNT)

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.

Permission is granted to members of INCITS, its technical committees, and their associated task groups to reproduce this document for the purposes of INCITS standardization activities without further permission, provided this notice is included. All other rights are reserved. Any duplication of this document for commercial or for-profit use is strictly prohibited.

T10 Technical Editor: Curtis E. Stevens
Seagate Technology
47488 Kato Road
Fremont, CA 94538
USA

Telephone: (949) 307-5050
Email: Curtis.Stevens@Seagate.com

Points of Contact

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

T10 Chair
William Martin
Samsung Semiconductor, Inc

T10 Vice-Chair
Curtis Ballard
Hewlett Packard Enterprise

Telephone: (408) 499-1839
Email: bill dot martin at samsung dot com

Telephone: (970) 898-6669
Email: curtis dot ballard at hpe dot com

T10 Web Site: <https://www.t10.org>

INCITS Secretariat
700 K Street NW Suite 600
Washington, DC 20001
USA

Telephone: (202) 737-8888
Web site: <https://www.incits.org>
Email: incits@itic.org

Information Technology Industry Council
Web site: <https://www.itic.org>

Purchase INCITS Standards
Web site: <https://www.incits.org/standards-information/purchase-standards-or-download-dpans>

SCSI / NVMe Translation

Secretariat
Information Technology Industry Council

Approved mm.dd.yy

American National Standards Institute, Inc.

ABSTRACT

This standard specifies a translation layer between SCSI and NVMe protocols. This translation layer is used by storage controllers to emulate objects in a SCSI logical unit using an NVMe device, providing capabilities defined by SCSI standards (e.g., the SCSI Block Commands (SBC-5), SCSI Primary Commands (SPC-6), and Zoned Block Commands (ZBC-2) standards). For the purposes of this standard, NVMe device capabilities are defined by the NVM Express family of standards.

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.

The patent statement goes here.

If no patents have been disclosed, then use the no disclosed.

If any patents have been disclosed, then use the patents disclosed statement.

Published by

American National Standards Institute

25 West 43rd Street 4th floor, New York, New York 10036-7422

Copyright © 20xx by Information Technology Industry Council (ITI). All rights reserved.

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without prior written permission of ITI, 700 K Street NW Suite 600, Washington, DC 20001.

Printed in the United States of America

Editor's Note 1: Make sure the copyright year is correct.

Revision Information

R.1 Revision SNTTr00 (2-May-2025)

This is revision 0 of this working draft.

R.2 Revision SNTTr01 (3-May-2025)f

- 1) Several of the paragraph formats have issues. Formats are being corrected in the Clause 11 file. The modified formats will be imported to the other files in Rev 2. formats include:
 - 1) TableCodeList (improper cell margins)
 - 2) TableFootnote (missing format info)
 - 3) Example (not indented)
 - 4)
- 2) Incorporated 24-060r4 - contains several translations for VPD pages
- 3) Incorporated 25-020r0 - Corrects the Mode Page Policy VPD page translation.
- 4) Incorporate 24-066r3

Contents

	Page
FOREWORD	x
INTRODUCTION	x
General	x
SCSI standards family	xi
 1 Scope	 1
 2 Normative References	 2
 3 Definitions, symbols, abbreviations, and conventions	 3
3.1 Definitions	3
3.2 Symbols and abbreviations	3
3.2.1 Abbreviations	3
3.2.2 Units	4
3.2.3 Symbols	4
3.2.4 Mathematical operators	4
3.3 Keywords	5
3.4 Editorial conventions	6
3.5 Numeric and character conventions	7
3.5.1 Numeric conventions	7
3.5.2 Units of measure	8
3.5.3 Byte encoded character strings conventions	8
3.5.4 Notation for command descriptions	9
3.5.5 Use of field names defined in ATA standards and specifications	9
3.6 Bit and byte ordering	10
3.7 Notation for procedure calls	11
 4 General	 13
4.1 General overview	13
4.2 NVMe capability detection	13
4.2.1 NVMe capability detection overview	13
4.2.2 DSM Supported	14
4.2.3 WZ Supported	14
4.2.4 IPI Formatted	15
 5 SCSI architecture	 17
5.1 Overview	17
5.2 Logical unit	18
5.3 Command descriptor block	19
5.3.1 Command descriptor block overview	19
5.3.2 Allocation length	19
5.3.3 CONTROL byte	19
5.4 Logical Block Length	19
 6 Command management model	 21
6.1 Overview	21
 7 Summary of SCSI / NVMe command mappings	 23
 8 SPC-5 command mapping	 25
8.1 INQUIRY command	25
8.1.1 Overview	25
8.1.2 Standard INQUIRY parameter data	26
 9 SBC-5 command mapping	 29

10 ZBC-2 command mapping	31
11 Parameters for SNT implementations	33
11.1 Overview	33
11.2 Diagnostic parameters	33
11.2.1 General Information	33
11.3 Log parameters	33
11.3.1 Overview	33
11.4 Mode parameters	34
11.4.1 General information	34
11.4.2 Overview	34
11.4.3 Mode parameter header	35
11.4.4 Mode parameter block descriptor fields	35
11.5 Vital product data parameters	36
11.5.1 Overview	36
11.5.2 Block Device Characteristics VPD page (i.e., B0h)	37
11.5.2.1 Block Device Characteristics VPD page (i.e., B0h) overview	37
11.5.2.2 MEDIUM ROTATION RATE field	38
11.5.3 Block Limits VPD page (i.e., B0h)	38
11.5.3.1 Block Limits VPD page (i.e., B0h) overview	38
11.5.3.2 MAXIMUM UNMAP LBA COUNT field	40
11.5.3.3 MAXIMUM WRITE SAME LENGTH field	41
11.5.4 Device Identification VPD page	41
11.5.5 Extended INQUIRY Data VPD page (i.e., 86h)	42
11.5.6 Logical Block Provisioning VPD page	44
11.5.7 Mode Page Policy VPD page (i.e., B7h)	45
11.5.7.1 Mode Page Policy VPD page (i.e., B7h) overview	45
11.5.7.2 Mode Page Policy Descriptor translation	45
11.5.8 Supported Block Lengths and Protection Types VPD page	45
11.5.8.1 Supported Block Lengths and Protection Types VPD page overview	45
11.5.8.2 Logical block length and protection types descriptor list	46
11.5.9 Supported VPD Pages VPD page	47
11.5.10 Unit Serial Number VPD page	47
12 Translation of NVMe errors to SCSI errors	49
12.1 Overview	49
13 SNT specific SCSI extensions	51
13.1 Overview	51
13.2 SNT specific SCSI extensions	51
13.3 SNT specific mode pages	51
13.4 SNT specific VPD pages	51
13.5 SNT specific security protocols	51
13.6 SNT specific log pages	51
Annex A (informative) Sample algorithms for splitting commands	52
A.1 TBD	52
Bibliography	54

Tables

	Page
Table 1 – Numbering conventions	7
Table 2 – Comparison of decimal prefixes and binary prefixes	8
Table 3 — Format for translated command field descriptions	9
Table 4 – Example of ordering of bits and bytes within a data dword	10
Table 5 – Example of ordering of bits and bytes within a data dword element	11
Table 6 – CONTROL byte fields	19
Table 7 – Summary of SCSI / NVMe command mapping	23
Table 8 – INQUIRY CDB field translations	25
Table 9 – INQUIRY parameter data translations	26
Table 10 — Summary of SCSI / ATA log page mapping	33
Table 11 — Summary of SCSI / ATA mode page mapping	34
Table 12 – VPD page translations	36
Table 13 – Block Device Characteristics VPD page field translations	37
Table 14 – MEDIUM ROTATION RATE field translation	38
Table 15 – Block Limits VPD page field translations	38
Table 16 – MAXIMUM UNMAP LBA COUNT field translation	40
Table 17 – MAXIMUM WRITE SAME LENGTH field translation	41
Table 18 – Extended INQUIRY Data VPD page field translations	42
Table 19 – Logical Block Provisioning VPD page field translations	44
Table 20 – Mode Page Policy VPD page field translations	45
Table 21 – Mode Page Policy Descriptor field translations	45
Table 22 – Supported Block Lengths and Protection Types VPD page field translations	45
Table 23 – Logical Block Length and Protection Types descriptor translation	46
Table 24 – Descriptor Example	47
Table 25 – Supported VPD Pages VPD page field translations	47

Figures

	Page
Figure 1 – Relationship of SNT to SCSI standards and NVMe specifications	xi
Figure 2 — Architectural overview of an SNTL between a SCSI application client and an NVMe device	17
Figure 3 — SNTL contained within a SCSI to NVMe protocol bridge	18
Figure 4 – SNTL contained within an NVME host	18

FOREWORD (This foreword is not part of American National Standard INCITS 584.)

This standard provides a common set of definitions and requirements to establish common behavior among implementations that emulate SCSI device behavior through the combined use of NVMe devices and a SCSI/NVMe Translation Layer (SNTL). The SNTL may reside in a host-based software or firmware, or it may reside in a separate component (e.g., a host bus adapter or external controller) with a separate processing unit to perform the translation. A SNTL and NVMe device combination may provide a functional subset of common SCSI capabilities. There is also a range of optional emulated SCSI capabilities that may be supported, depending on the capabilities of the SNTL.

This standard defines SNTL capabilities in terms of SCSI capabilities as defined by the applicable SCSI standards and working drafts, and defines the elements and use of the NVMe protocol to provide those SCSI capabilities and services in a consistent manner among SNT implementations that are implemented as described by this standard.

Requests for interpretation, suggestions for improvement and addenda, or defect reports are welcome. They should be sent to the INCITS Secretariat, InterNational Committee for Information Technology Standards, Information Technology Industry Council, 700 K Street NW Suite 600, Washington, DC 20001.

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 it approved this standard, INCITS had the following members:

INCITS Technical Committee T10 on SCSI Storage Interfaces, which developed and reviewed this standard, had the following members:

William Martin, Chair
Curtis Ballard, Vice-Chair
Curtis Stevens, Secretary

Organization Represented

Name of Representative

INTRODUCTION**General**

The SCSI / NVMe Translation (SNT) standard is divided into the following clauses and annexes:

Clause 1 defines the scope of this standard.

Clause 2 enumerates the normative references that apply to this standard.

Clause 3 describes the definitions, symbols, abbreviations, and notation conventions used in this standard.

Clause 4 describes the general framework for defining elements of translation between SCSI and NVMe protocols.

Clause 5 describes elements of SCSI / NVMe Translation that relate to the SCSI architecture model.

Clause 6 describes the mapping of command management functions in the SNTL layer.

Clause 7 provides a summary of SCSI commands mapped to NVMe in this standard.

Clause 8 describes the mapping of SCSI Primary Commands to NVMe.

Clause 9 describes the mapping of SCSI Block Commands to NVMe and Zoned Block Commands to NVMe.

Clause 10 describes the mapping of diagnostic parameter, mode page, log page, and VPD page information to selected NVMe protocol elements.

Clause 11 describes error reporting and sense data conventions for SCSI / NVMe Translation.

Clause 12 describes SCSI commands and mode pages to support SCSI / NVMe Translation.

SCSI standards family

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

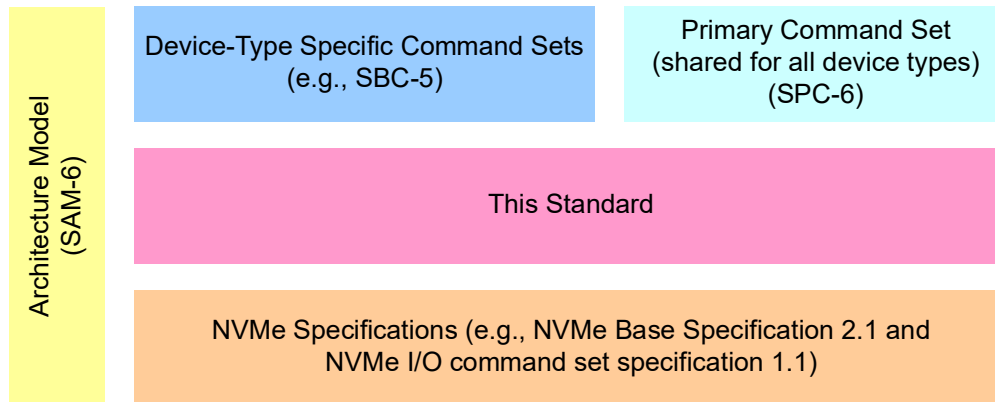


Figure 1 – Relationship of SNT to SCSI standards and NVMe specifications

The roadmap 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.

**American National Standard
for Information Technology -**

SCSI / NVMe Translation (SNT)

1 Scope

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.

This standard defines the protocol requirements of the SCSI / NVMe Translation Layer (SNTL) to allow conforming SCSI / NVMe translating components to interoperate with NVMe devices, SCSI transports, and SCSI application layers. The SNTL covers a range of implementations that use NVMe devices to emulate the behavior of SCSI devices as viewed by the SCSI application layer. The primary focus of this standard is to define SCSI / NVMe Translation for an NVMe device (see 3.1.5). Where this standard does not define a translation, then the applicable standard should be used.

Where possible, this standard defines SCSI / NVMe Translation in a manner that is consistent with the SAM-6, SPC-6, SBC-5, and ZBC-2 standards. In some instances, the defined function of an NVMe device is different from corresponding functions defined for SCSI target devices (e.g., many NVMe devices provide no means to abort a single NVMe queued command). The translation defined in this standard, in such cases, may not be consistent with other SCSI standards. However, in such cases, this standard specifies the expected behavior, and in what manner it is inconsistent with the behavior specified in other SCSI standards.

The objective of this standard is to allow an interoperable set of SCSI functions while minimizing the complexity of the SNTL and preserving compatibility with existing SCSI application clients.

The objectives of the SNTL are:

- a) to provide host computers with device independence with respect to the NVMe devices and with respect to various implementations of the translation layer used to emulate the behavior of SCSI target devices;
- b) to define common features and functions representing a subset of the capabilities available in SCSI devices that apply to SCSI / NVMe Translation implementations;
- c) to define common methods to manage aspects of NVMe devices that do not map to previously defined features and functions of SCSI, with provision made for the addition of special features and functions; and
- d) to provide consistent means for discovery and control of optional SCSI features that may or may not be emulated in SCSI / NVMe translator implementations. These means are provided by specifying how transport specific features and functions are represented in a manner consistent with management of devices in a SCSI domain.

2 Normative References

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

ISO/IEC 14776-416, *Information technology – Small Computer System Interface (SCSI) – Part 416: SCSI Architecture Model - 6 (SAM-6)*

INCITS 566, *Information Technology – SCSI Primary Commands - 6 (SPC-6)*

INCITS 571, *Information Technology – SCSI Block Commands - 6 (SBC-5)*

INCITS 579, *Information Technology – Zoned Block Commands - 3 (ZBC-3)*

For information on the current status of the listed documents, or regarding availability, contact the indicated organization.

NVM Express 1.4 (NVMe 1.4). Date¹

NVM Express Base Specification 2.1 (NVMBASE), August 5th, 2024²

NVM Express Command Set Specification Revision 1.1 (NVMCS), August 5th, 2024³

NVM Express Zoned Namespace Command Set Specification Revision 1.2, August 5th, 2024⁴

NOTE 1 - For more information on NVM Express organization, [see www.nvmexpress.org](http://www.nvmexpress.org).

-
1. Information TBP
 2. Information TBP
 3. Information TBP
 4. Information TBP

3 Definitions, symbols, abbreviations, and conventions

3.1 Definitions

3.1.1 allocation length

value in the ALLOCATION LENGTH field of a CDB that specifies the maximum number of bytes that an application client has allocated in the Data-In Buffer and is used to limit the maximum amount of variable length data returned to an application client

Note 1 to entry: See SPC-7

Note 2 to entry: Examples of variable length data include mode data, log data, and diagnostic data.

3.1.2 byte

8-bit construct

3.1.3 command descriptor block (CDB)

structure used to communicate a command from a SCSI application client to a SCSI device server

Note 1 to entry: See SAM-6

3.1.4 field

group of one or more contiguous bits

3.1.5 NVMe device

NVM subsystem that complies with the NVM Express family of specifications

3.1.6 SCSI device

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

3.2 Symbols and abbreviations

3.2.1 Abbreviations

Abbreviations used in this standard:

Abbreviation	Meaning
B	byte
CDB	command descriptor block (see 3.1.3)
LSB	least significant bit
LUN	logical unit number
MSB	most significant bit
NVMeBASE	NVM Express Base Specification Revision 2.1 (see clause 2)
NVMeCS	NVM Express NVM Command Set Specification Revision 1.1 (See clause 2)
NVMe	NVM Express

SBC-5 SCSI Block Commands-5 (see clause)
 SPC-6 SCSI Primary Commands-6 (see clause)
 SNT SCSI / NVMe Translation
 SNTL SCSI / NVMe Translation Layer

3.2.2 Units

Units used in this standard:

Units Meaning

μs microsecond (i.e., 10^{-6} seconds)
 ms millisecond (i.e., 10^{-3} seconds)
 ns nanosecond (i.e., 10^{-9} seconds)
 s second (unit of time)

3.2.3 Symbols

Symbols used in this standard:

Symbols	Meaning
®	registered trademark
™	trademark

3.2.4 Mathematical operators

Mathematical operators used in this standard:

Mathematical Operators	Meaning
^ or XOR	exclusive logical OR
×	multiplication
/	division
±	plus or minus
≈	approximately
~	approximately equal to
+	add
-	subtract
< or LT	less than
≤ or LE	less than or equal to
= or EQ	equal
≠ or NE	not equal
> or GT	greater than
≥ or GE	greater than or equal to
v	the absolute value (i.e., magnitude) of v
∈	set membership
MIN (x, y, ...)	minimum of a list of values

3.3 Keywords

3.3.1 invalid

keyword used to describe an illegal or unsupported bit, byte, word, field, or code value

Note 1 to entry: Receipt by a device server of an invalid bit, byte, word, field, or code value shall be reported as an 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

3.3.4 may or may not

keyword that indicates flexibility of choice with no implied preference

Note 1 to entry: Significant uses of "may or may not" occur in descriptions where attention is being drawn to the "may not" case.

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

Note 1 to entry: 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 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

Note 1 to entry: A reserved bit, byte, word, or field shall be set to zero, or in accordance with a future extension to this standard.

Note 2 to entry: Recipients are not required to check reserved bits, bytes, words, or fields for zero values.

Note 3 to entry: Receipt of reserved code values in defined fields shall be reported as error.

3.3.9 restricted

keyword referring to bits, bytes, words, and fields that are set aside for other identified standardization purposes

Note 1 to entry: 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

Note 1 to entry: 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

3.3.12 vendor specific

something (e.g., a bit, field, code value) that is not defined by this standard

Note 1 to entry: Specification of the referenced item is determined by the SCSI device vendor and may be used differently in various implementations.

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 (e.g., **Execute Command**). Names of arguments are denoted by capitalizing each word in the name (e.g., Sense Data is the name of an argument in the **Execute Command** procedure call). For more information on procedure calls see 3.7.

Quantities having a defined numeric value are identified by large capital letters (e.g., CHECK CONDITION). Quantities having a discrete but unspecified value are identified using small capital letters. (e.g., TASK COMPLETE, indicates a quantity returned by the **Execute Command** procedure call). 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) various forms of red such as:
 - 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.

Lists are associated with an introductory paragraph or phrase, and are numbered relative to that paragraph or phrase (i.e., all lists begin with an a) or 1) entry).

If a conflict arises between text, tables, or figures, then the order of precedence to resolve the conflicts is:

- 1) text;
- 2) tables; and

3) figures.

Not all tables or figures are fully described in the text.

Notes and examples do not constitute any requirements.

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 comprised of only the 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 organizational boundaries (e.g., 00010101 11001110b, 00010101_11001110b, 0 0101 1010b or 0_0101_1010b).

A hexadecimal number is represented in this standard by any sequence of digits comprised of only the Arabic numerals 0 to 9 and/or the upper-case English letters A to 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 organizational boundaries (e.g., 3456FDCA 84BD5E7Ah, 3456FDCA_84BD5E7Ah, B FD8C FA23h, or B_FD8C_FA23h).

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

Variables (i.e., alphanumeric names that represent values in computations and other statements) are represented in the same San-serif font as other information in this standard.

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

A decimal number represented in this standard with an overline over one or more digits following the decimal point is a number where the overlined digits are infinitely repeating (e.g., 666. $\overline{6}$ means 666.666 666... or 666 2/3, and 12. $\overline{142\ 857}$ means 12.142 857 142 857... or 12 1/7).

A range of numeric values is represented in this standard in the form “a to z”, where a is the first value included in the range, all values between a and z are included in the range, and z is the last value included in the range (e.g., the representation “0h to 3h” includes the values 0h, 1h, 2h, and 3h).

3.5.2 Units of measure

This standard represents values using both decimal units of measure and binary units of measure. Values are represented by the following formats:

- a) for values based on decimal units of measure:
 - 1) numerical value (e.g., 100);
 - 2) space; and
 - 3) prefix symbol and unit:
 - 1) decimal prefix symbol (e.g., M) (see table 2); and
 - 2) unit abbreviation;
- and
- b) for values based on binary units of measure:
 - 1) numerical value (e.g., 1 024);
 - 2) space; and
 - 3) prefix symbol and unit:
 - 1) binary prefix symbol (e.g., Gi) (see table 2); and
 - 2) unit abbreviation.

Table 2 compares the prefix, symbols, and power of the binary and decimal units.

Table 2 – Comparison of decimal prefixes and binary prefixes

Decimal			Binary		
Prefix name	Prefix symbol	Power (base-10)	Prefix name	Prefix symbol	Power (base-2)
kilo	k	10^3	kibi	Ki	2^{10}
mega	M	10^6	mebi	Mi	2^{20}
giga	G	10^9	gibi	Gi	2^{30}
tera	T	10^{12}	tebi	Ti	2^{40}
peta	P	10^{15}	pebi	Pi	2^{50}
exa	E	10^{18}	exbi	Ei	2^{60}
zetta	Z	10^{21}	zebi	Zi	2^{70}
yotta	Y	10^{24}	yobi	Yi	2^{80}

3.5.3 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' are to be encoded is the same as writing out the following sequence of byte values: 53h 43h 53h 49h 20h 64h 65h 76h 69h 63h 65h.

3.5.4 Notation for command descriptions

The description of each command begins with a subclause describing the general method applied in translating the SCSI command to the corresponding ATA commands, along with any constraints and special considerations that may apply to the translation applied.

The subclause describing the general translation method for each command contains a table formatted as shown in table 3 with two columns as follows:

- a) the first column lists each of the fields in the SCSI CDB (see SPC-5 and SBC-4); and
- b) the second column is either a brief description of the corresponding ATA features and functions used to implement the identified SCSI field, or a reference to a subsequent subclause containing a more lengthy description of the method of emulation or implementation.

Table 3 — Format for translated command field descriptions

Field	Description
IMPLEMENTED OR EMULATED	A brief identification of the corresponding ATA features and functions, or a paragraph reference if there are special considerations that need to be applied in the use of the corresponding ATA features and functions that require a separate paragraph of description.
SUMMARY EMULATED	Summary field with more detailed structure.
UNSPECIFIED	Unspecified

Tables listing fields in mode pages have an additional column that defines whether the field is changeable or not.

3.5.5 Use of field names defined in ATA standards and specifications

This standard discusses fields and values defined in other standards and specifications (e.g., the ATA8-APT standard, the ATA8-AST standard, the ACS-5 standard, and the ATA8-AAM standard) developed by T13 and the SATA-3.5a specification. Such fields and values discussed in this standard are shown using the same notation conventions used in the standards where those fields and values are defined.

When this standard uses terms defined in ATA standards or the SATA-3.5a specification, the following conventions apply:

- a) the names of abbreviations, commands, and acronyms used as signal names are in all uppercase (e.g., READ FPDMA QUEUED);
- b) names of device registers, fields in data structures, and other defined terms are in small upper-case letters (e.g., FEATURES field);
- c) the expression “word n” or “bit n” shall be interpreted as indicating the content of word n or bit n;
- d) bit names are shown in small uppercase letters (e.g., REPORT ZONES EXT SUPPORTED bit); and
- e) bits n:m denotes a set of bits, for example, bits 7:0.

Editor's Note 1: The next few subclauses define conventions for UML diagrams, state machines, and procedure calls. Any of these may be deleted if the described convention is not used in the draft standard.

3.6 Bit and byte ordering

In this standard, data structures may be defined by a table. A table defines a complete ordering of elements (i.e., bits, bytes, fields, and dwords) within the structure. The ordering of elements within a table does not in itself constrain the order of storage or transmission of the data structure, but in combination with other normative text in this standard, the ordering of elements within a table may constrain the order of storage or transmission of the structure.

In a table, any element that is presented in a row above another element in a lower row is more significant than the lower element, and any element presented to the left of another element in the same row is more significant than the element to the right.

If a table shows bit numbering (see table 4), the least significant bit (LSB) is numbered 0 and each more significant bit has the next greater number than the immediately less significant bit. If a table shows numbering of bytes or characters (see table 5), the most significant byte or character is represented at the lowest number and each less significant byte or character has the next greater number than the immediately more significant byte.

In a field in a table consisting of more than one bit that contains a single value (e.g., a number), the least significant bit (LSB) is shown on the right and the most significant bit (MSB) is shown on the left (e.g., in a byte, bit 7 is the MSB and is shown on the left, bit 0 is the LSB and is shown on the right). The MSB and LSB are not labeled if the field consists of eight or fewer bits. The MSB and LSB are labeled if the field consists of more than eight bits and has no internal structure defined.

In a field in a table consisting of more than one byte that contains multiple fields each with their own values (e.g., a descriptor), there is no MSB and LSB of the field itself and thus there are no MSB and LSB labels. Each individual field has an MSB and LSB, but they are not labeled.

In a field containing a text string (e.g., ASCII or UTF-8), only the MSB of the first character and the LSB of the last character are labeled.

Multiple byte fields are represented with only two rows, with the non-sequentially increasing byte number denoting the presence of additional bytes.

A data dword consists of 32 bits. Table 4 shows a data dword containing a single value, where the MSB is on the upper left in bit 31 and the LSB is on the lower right in bit 0.

Table 4 – Example of ordering of bits and bytes within a data dword

Bit Byte	7	6	5	4	3	2	1	0
0	(MSB) Bit 31	Bit 30	Bit 29	Bit 28	Bit 27	Bit 26	Bit 25	Bit 24
1	Bit 23	Bit 22	Bit 21	Bit 20	Bit 19	Bit 18	Bit 17	Bit 16
2	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
3	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
Note - The Bit x labels in the individual table cells are for reference only and should not appear within tables that use this element format.								

Table 5 shows a data dword containing four one-byte elements, where byte 0 (the first byte) is on the top and byte 3 (the fourth byte) is on the bottom. Each byte has an MSB on the left and an LSB on the right.

Table 5 – Example of ordering of bits and bytes within a data dword element

Bit Byte	7	6	5	4	3	2	1	0
0	First byte							
	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
1	Second byte							
	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
2	Third byte							
	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
3	Fourth byte							
	Bit 7 (MSB)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0 (LSB)
Note - The Bit x labels in the individual table cells and the xx byte labels in the individual bytes are for reference only and should not appear within tables that use these element formats. In this example the MSB and LSB labels are for reference only. However, they may appear in multi-byte fields as described in this subclause.								

3.7 Notation for procedure calls

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

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

Where:

Result	A single value representing the outcome of the procedure or function.
Procedure Name	A descriptive name for the function to be performed.
IN (Input-1, Input-2, ...)	A comma-separated list of names identifying caller-supplied input data objects.
OUT (Output-1, Output-2, ...)	A comma-separated list of names identifying output data objects to be returned by the procedure.
[...]	Brackets enclose optional or conditional parameters and arguments.

This notation allows arguments to be specified as inputs and outputs. An interface between entities may require only inputs. If a procedure call has no output arguments, the word OUT, preceding comma, and associated pair of balanced parentheses are omitted.

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.

4 General

4.1 General overview

This standard defines a SCSI / NVMe Translation Layer (i.e., the SNTL) that provides a method for a SCSI application layer (see SAM-6) to access NVMe devices by representing NVMe devices as SCSI peripheral devices.

Implementations of SCSI / NVMe Translation may provide varying levels of SCSI functionality.

EXAMPLE 1 - The SNTL may provide a level of SCSI emulation that is indistinguishable from native SCSI devices in terms of reported capabilities. Such SNTL implementations need little guidance from this standard to effect interoperability since other SCSI protocol standards define all that is required to establish interoperability.

EXAMPLE 2 - The SNTL may implement a subset of SCSI, have limited or no capability to maintain persistent information about the characteristics or state of the emulated SCSI device, have limited capability to manage device state information that carries forward from one command to the next, and maintain little or no capability to coordinate between multiple commands outstanding at a time. The characteristics and behavior of the underlying NVMe devices in these minimal implementations of the SNTL are expected to be more visible to the SCSI application clients.

This standard provides a set of definitions, conventions, and guidelines for:

- a) the consistent reporting by the SNTL of capabilities of emulated SCSI devices;
- b) the consistent observed behavior for SCSI operations; and
- c) the consistent identification of the attached devices by the application clients.

These provisions allow application clients to observe consistent behavior whether or not the application clients recognize the presence of a SNTL in a system.

By defining expected behavior in terms of the SCSI commands received, corresponding activity in the NVMe domain, and expected SCSI responses based on the results of activity in the NVMe domain this standard eliminates:

- a) incompatibility between legacy SCSI / NVMe Translation implementations; and
- b) SCSI application client / ATA device interdependence.

This standard refers to behaviors for SCSI devices defined in SBC-5, ZBC-2, and SPC-6. Unless otherwise specified, any behaviors that are optional in SBC-5 or SPC-6 are optional for devices implementing SCSI / NVMe Translation.

If the SNTL receives a SCSI request specifying any value in any field of the CDB that the SNTL does not support then, unless otherwise specified in the description of the command, the SNTL shall terminate the SCSI command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB (see SPC-6).

If the SNTL receives a SCSI request specifying any value in any field of the parameter data that the SNTL does not support then, unless otherwise specified in the description of the parameter, the SNTL shall terminate the SCSI command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN PARAMETER LIST (see SPC-6).

4.2 NVMe capability detection

4.2.1 NVMe capability detection overview

This clause defines how NVMe capabilities are detected and result in internal variables used by the SNTL for translations.

4.2.2 DSM Supported

Editor's Note 2: NVMe 1.4:

Bit 2 if set to '1', then the controller supports the Dataset Management command. If cleared to '0', then the controller does not support the Dataset Management command.

NVMe 2.0:

Bit 2 if set to '1', then the controller supports the NVM Command Set Dataset Management command and limits, if any, on controller support of the Dataset Management command are indicated by non-zero values in the Dataset Management Ranges Limit (DMRL) field, the Dataset Management Size Limit (DMSL) field and the Dataset Management Range Size Limit (DMRSL) field. If cleared to '0', then controller support of the NVM Command Set Dataset Management command is indicated by a non-zero data size limit in the DMRL, DMSL, and DMRSL fields.

DSM Supported is an internal variable that indicates if the NVMe device supports the Data Set Management command.

DSM Supported shall be set to true:

- a) if, in the NVMe Identify Controller data structure (CNS 01h) ONCS field (see NVMeBASE), the NVMDSMSV bit is set to 1b; or
- b) if, in the NVMe I/O Command Set Specific Identify Controller data structure (CNS 06h, CSI 00h), DMRL field, DMSL field, or DMRSL field (see NVMCS) is set to a non-zero value;

otherwise, DSM Supported shall be set to false.

4.2.3 WZ Supported

Editor's Note 3: NVMe 1.4:

Bit 3 if set to '1', then the controller supports the Write Zeroes command. If cleared to '0', then the controller does not support the Write Zeroes command.

NVMe 2.0:

Bit 3 if set to '1', then the controller supports the NVM Command Set Write Zeroes command and the Write Zeroes Size Limit (WZSL) field indicates the recommended maximum data size for Write Zeroes commands. If cleared to '0', then controller support of the NVM Command Set Write Zeroes command is indicated by a non-zero data size limit in the WZSL field.

WZ Supported is an internal variable that indicates if the NVMe device supports the Write Zero command.

WZ Supported shall be set to true:

- a) if, in the NVMe Identify Controller data structure (CNS 01h) ONCS field (see NVMeBASE), the NVMWZSV bit is set to 1b; or
- b) if, the NVMe I/O Command Set Specific Identify Controller data structure (CNS 06h, CSI 00h) WZSL field (see NVMCS) is set to a non-zero value;

otherwise, WZ Supported shall be set to false.

4.2.4 IPI Formatted

IPI Formatted is an internal variable that indicates if the NVMe namespace is formatted with Interleaved Protection Information (i.e., Protection Information that is transferred at the end of the logical block data).

IPI Formatted shall be set to true:

- a) if the NVMe Identify Namespace data structure (CNS 00h) DPS field (see NVMCS) is not set to 000b;
and
- b) if, in the NVMe Identify Namespace data structure (CNS 00h) FLBAS field (see NVMCS), the MTELBA bit is set to 1b;

otherwise, IPI Formatted shall be set to false.

5 SCSI architecture

5.1 Overview

Clause 5 defines SCSI / NVMe translation of features and functions that impact the representation of the domains defined in SAM-6 and NVMeBASE. Figure 2 shows an SNTL providing a communication path between a SCSI application client and an NVMe device.

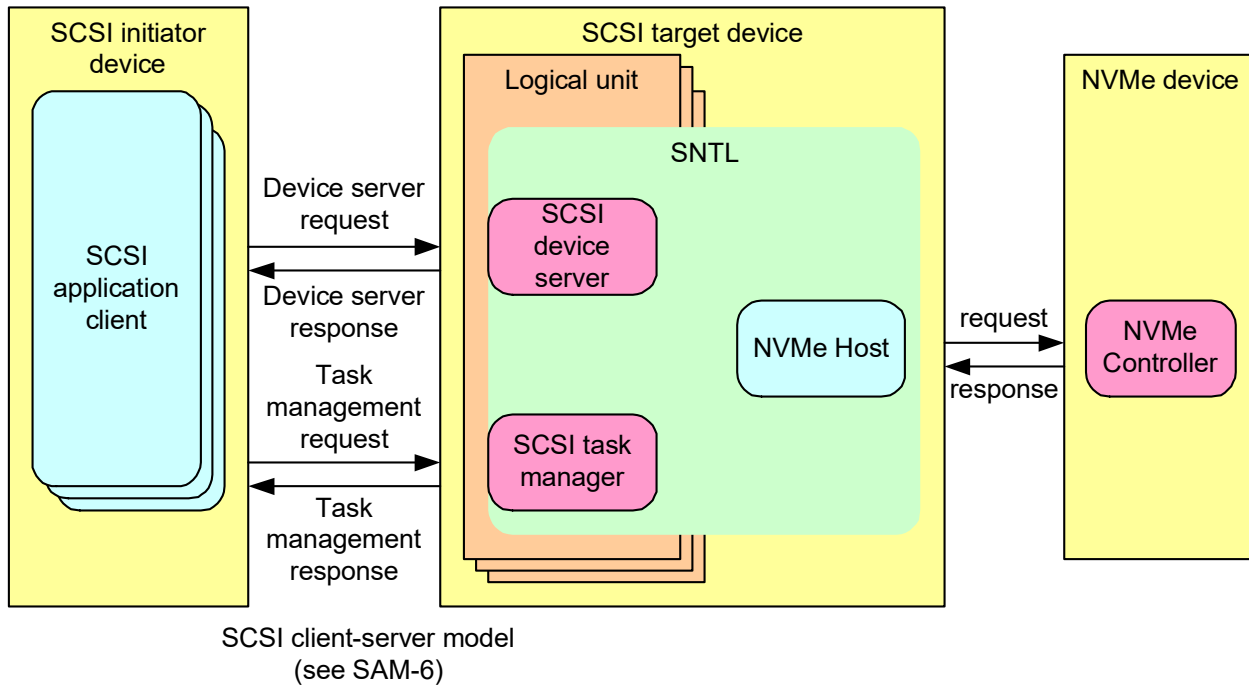


Figure 2 — Architectural overview of an SNTL between a SCSI application client and an NVMe device

The SNTL provides the communication path between a SCSI application client and an NVMe device by:

- emulating a SCSI logical unit;
- integrating an NVMe host; and
- providing the translation that links them together.

This standard defines SCSI / NVMe translation using SCSI and NVMe command sets. This standard does not define the mapping of transport capabilities as defined at the SCSI transport protocol layer and the NVMe protocol interconnect layer.

EXAMPLE 1 - An implementation utilizing a SNTL may include a SCSI transport. An SNTL may appear in different configurations as shown in figure 3 and figure 4. Figure 3 shows an SNTL contained within a SCSI to NVMe protocol bridge, where the NVMe device is being accessed by an NVMe host port and the SNTL is being accessed with a SCSI target port using a SCSI transport protocol.

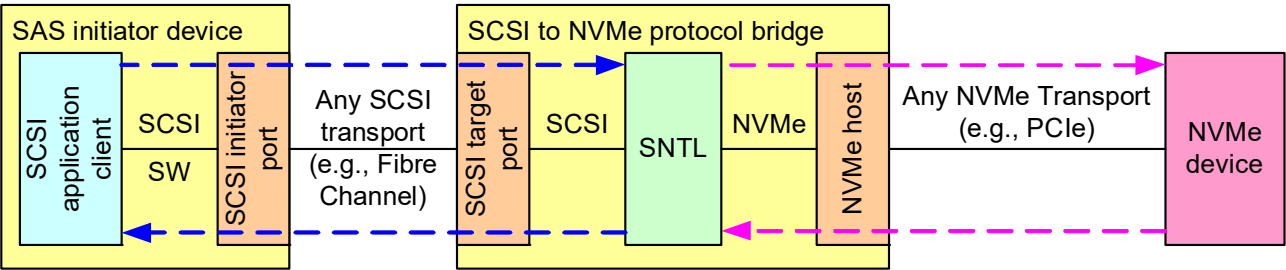


Figure 3 — SNTL contained within a SCSI to NVMe protocol bridge

EXAMPLE 2 - Figure 4 shows an NVMe HBA directly connected to an NVMe device. The SNTL provides SCSI transport protocol layer services to a SCSI application client in accordance with SAM-6.

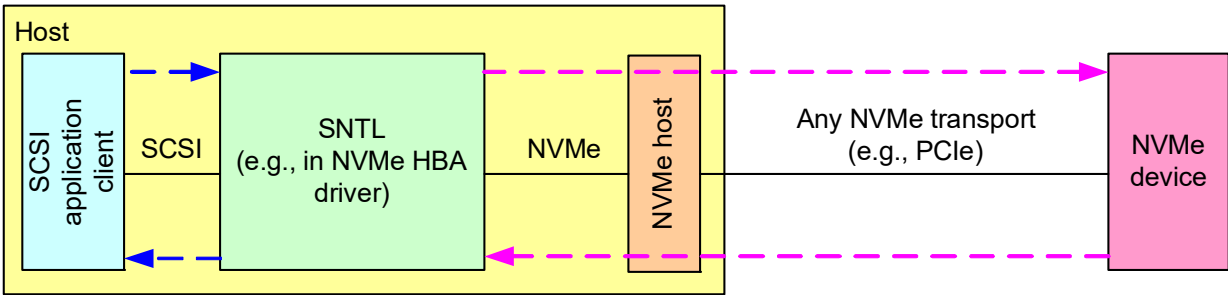


Figure 4 – SNTL contained within an NVME host

5.2 Logical unit

Editor's Note 4: This does not exist in SAT, does it go here or the command management model?

Editor's Note 5: This needs more scrutiny

A logical unit (see SAM-6) within a SCSI target device is conceptually similar to an NVMe namespace. The SNTL shall use the SCSI TARGET OR LUN field in the SCSI single level LUN structure for all NSID fields in the applicable NVMe commands by adding one to the specified SCSI LUN. Implementations of the SNTL may have further restrictions on the supported SCSI LUN and NSID values.

Editor's Note 6: Needs updating to support fabrics

...

5.3 Command descriptor block

5.3.1 Command descriptor block overview

The translations of common fields in a CDB (see SAM-6) that are used in multiple commands are described in 5.3.

5.3.2 Allocation length

The ALLOCATION LENGTH field (see SPC-7) specifies the maximum number of bytes returned into a memory buffer. If the command has more data to return, the SNTL shall not return more bytes than the value specified in this field.

5.3.3 CONTROL byte

Table 6 describes SNTL translation of the CDB CONTROL byte. See SAM-6 for CONTROL byte details.

Table 6 – CONTROL byte fields

Field	Description
Vendor specific	The SNTL may use this field for vendor specific purposes.
NACA	If this bit is set to one, then the SNTL shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.

5.4 Logical Block Length

The Logical Block Length for SCSI is the number of user data bytes in a logical block. A given NVMe namespace may be formatted to one of up to 16 LBA formats supported by the NVMe Controller. The current LBA format is indicated by the NVMe Identify Namespace data structure (CNS 00h) FLBAS field (see NVMCS). This value is then used as an index to the current NVMe LBA Format data structure in the NVMe Identify Namespace data structure (CNS 00h) (see NVMCS), containing a LBADS field. That LBADS field is then converted from a power-of-2 value into the resulting Logical Block Length ($1 \ll \text{LBADS}$).

6 Command management model

6.1 Overview

A SNTL may support the full task management model or the basic task management model as well as specific features of the task management model (e.g., SIMPLE and ORDERED task attributes) depending on the task management capabilities of the SNTL.

Editor's Note 7: This needs to be developed after r0 is published

7 Summary of SCSI / NVMe command mappings

In the event of a discrepancy between the contents of this clause and the description of individual commands, description of individual commands shall apply.

Clause 7, clause 8, clause 9, and clause 10 describe the SCSI to NVMe command mapping for NVMe devices emulating a SCSI logical unit with a peripheral device type of 00h (i.e., direct access block device) or 14h (i.e., host managed zoned device).

Table 12 lists the SCSI / ATA command mappings defined in this standard. A SNTL may implement commands defined in SPC-6, SBC-5, and ZBC-2, but not listed in table 12. Translation of commands not listed in table 12 is vendor specific. If a command is not implemented by the SNTL, then the SNTL shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID COMMAND OPERATION CODE.

Table 7 – Summary of SCSI / NVMe command mapping

SCSI command	Reference
INQUIRY	8.1

8 SPC-5 command mapping

8.1 INQUIRY command

8.1.1 Overview

[Editor's Note 8: Do we need a definition of NVM subsystem?](#)

The INQUIRY command requests general information about a logical unit and target device. The INQUIRY command and selected VPD pages is emulated using information from the NVM subsystem data structures.

Table 8 shows the translation for the fields of the INQUIRY CDB.

Table 8 – INQUIRY CDB field translations

Field	Description
OPERATION CODE	Set to 12h
EVDP	The SNTL shall, if this bit is set to: <ol style="list-style-type: none"> 1) 0b and the PAGE CODE field is not set to 00h, then terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB; 2) 0b and the PAGE CODE field is set to 00h, then return standard INQUIRY parameter data (see 8.1.2); and 3) 1b, then return the vital product data specified by the PAGE CODE field.
PAGE CODE	If the EVDP field is set to 1b, the SNTL returns the valid product data for a supported page code as described in 11.5 or other vendor specific VPD pages. If this field is set to an unsupported page code, then the SNTL shall terminate the command with CHECK CONDITION status with the sense key set to ILLEGAL REQUEST and the additional sense code set to INVALID FIELD IN CDB.
ALLOCATION LENGTH	See 5.3.2.
CONTROL	See 5.3.3.

8.1.2 Standard INQUIRY parameter data

The SNTL shall return standard INQUIRY parameter data as described in SPC-7.

The SNTL shall validate the LUN value associated with the command. If any byte in the LUN, other than the TARGET OR LUN field in a Single level LUN structure (see SAM-6), is set to a non-zero value, or the LUN value is not associated with a valid NVMe namespace (see 5.2), then the PERIPHERAL QUALIFIER and PERIPHERAL DEVICE TYPE shall be set as specified in Table 3 for an invalid LUN.

[Editor's Note 9: Needs updating to support fabrics](#)

Table 9 describes the translation of fields in the standard INQUIRY parameter data.

Table 9 – INQUIRY parameter data translations

Field	Description
PERIPHERAL QUALIFIER	Shall be set to 000b if the command is for a valid LUN (i.e., the LUN is mapped to an NVMe namespace), otherwise shall be set to 011b.
PERIPHERAL DEVICE TYPE	Shall be set to 00h if the command is for a valid LUN (i.e., the LUN is mapped to an NVMe namespace), otherwise shall be set to 1Fh.
RMB	Shall be set to 0b (i.e., medium is not removable).
LU_CONG	Shall be set to 0b (i.e., logical unit is not part of a logical unit conglomerate).
HOT PLUGGABLE	Shall be set to 00b (i.e., no information is provided).
VERSION	This field shall be set to the version of SPC to which the SNTL complies (e.g., 0Eh indicates SPC-7). The value reported shall indicate support for SPC-7 or a subsequent version.
NORMACA	Shall be set to 0b (i.e., NACA bit and ACA task attribute not supported).
HiSUP	Shall be set to 1b.
RESPONSE DATA FORMAT	Shall be set to 2h.
ADDITIONAL LENGTH	Shall be set to the number of bytes that follow the ADDITIONAL LENGTH field.
SCCS	Shall be set to 0b (i.e., does not contain a storage array controller).
TPGS	Shall be set to 00b (i.e., no support for asymmetric logical unit access).
3PC	Shall be set to 0b (i.e., third-party copy not supported).
PROTECT	Shall be set to 0b (i.e., does not support protection information): <ul style="list-style-type: none"> a) if the SNTL does not support protection information; or b) if the NVMe Identify Namespace data structure (CNS 00h) DPC field (see NVMCS): <ul style="list-style-type: none"> A) PIT1S bit (i.e., PI Type 1 supported) is set to 0b; B) PIT2S bit (i.e., PI Type 2 supported) is set to 0b; and C) PIT3S bit (i.e., PI Type 3 supported) is set to 0b; otherwise this bit shall be set to 1b (i.e., supports protection information).
ENC SERV	Shall be set to 0b (i.e., no embedded enclosure services).
VS	The SNTL may return vendor specific data, otherwise this bit shall be set to 0b.
MULTIP	Shall be set to the value of the MPORTS bit in the NVMe Identify Controller data structure (CNS 01h) CMIC field (see NVMBASE) (i.e., multiple NVM subsystem ports).

Table 9 – INQUIRY parameter data translations (Continued)

Field	Description
CMDQUE	Shall be set to 1b.
T10 VENDOR IDENTIFICATION	Shall be set to "NVMe ^{١١١١} ". Note: An ASCII space character (i.e., 20h) is represented by the symbol '٠'.
PRODUCT IDENTIFICATION	Shall be set to the first 16 ASCII characters of the NVMe Identify Controller data structure (CNS 01h) MN field (see NVMBASE) (i.e., Model Number).
PRODUCT REVISION LEVEL	Shall be set to the last four contiguous bytes that are not used for padding (i.e., ASCII space characters, 20h) of the NVMe Identify Controller data structure (CNS 01h) FR field (see NVMBASE) (i.e., Firmware Revision).
Vendor Specific	The SNTL may return vendor specific data, otherwise all bytes shall be set to 00h.
VERSION DESCRIPTOR 1 to VERSION DESCRIPTOR 8	The SNTL provides support for VERSION DESCRIPTORS as follows: a) shall include the SCSI Architecture Model standard (e.g., 00C0h/SAM-6); b) shall include the SCSI Primary Commands standard (e.g., 0700/SPC-7); c) shall include the SCSI Block Commands standard (e.g., 0720h/SBC-6); and d) may include others.
Vendor specific parameters	The SNTL may return vendor specific data. If no vendor specific data is returned, the SNTL should not return this field.

9 SBC-5 command mapping

...

10 ZBC-2 command mapping

...

11 Parameters for SNT implementations

11.1 Overview

Parameters for all device types are defined in this clause as follows:

- a) diagnostic parameters are defined in 11.2;
- b) log parameters are defined in 11.3;
- c) mode parameters are defined in 11.4; and
- d) vital product data parameters are defined in 11.5.

11.2 Diagnostic parameters

11.2.1 General Information

SCSI diagnostic parameters provide a mechanism to initiate diagnostic functions and return results for those diagnostic functions. The SEND DIAGNOSTIC command is used to initiate diagnostic functions and the RECEIVE DIAGNOSTIC RESULTS command is used to obtain results from those functions.

11.3 Log parameters

[Editor's Note 10: This is the list of translated log pages from SAT-6, no translations are defined yet.](#)

11.3.1 Overview

The SNTL translations for log parameters are listed in table 10.

Table 10 — Summary of SCSI / ATA log page mapping (part 1 of 2)

SCSI log page name	Page code	Subpage code	Reference
Application Client	0Fh	00h	
Background Scan	15h	00h	
Command Duration Limits Statistics	19h	21h	
General Statistics and Performance	19h	00h	
Informational Exceptions	2Fh	00h	
Pending Defects	15h	01h	
Read Error Counters	03h	00h	
Self-Test Results	10h	00h	
Solid State Media	11h	00h	
Start-Stop Cycle Counter	0Eh	00h	

Table 10 — Summary of SCSI / ATA log page mapping (part 2 of 2)

SCSI log page name	Page code	Subpage code	Reference
Supported Log Pages	00h	00h	
Supported Log Pages and Subpages	00h	FFh	
Temperature	0Dh	00h	
Zoned Block Device Statistics	14h	01h	
All others			Unspecified

11.4 Mode parameters

11.4.1 General information

SCSI mode parameters provide a mechanism to set operating parameters for SCSI devices and logical units. The MODE SENSE command obtains operating parameters and the MODE SELECT command sets operating parameters. This standard does not define the content of most operating parameters defined in mode pages due to lack of equivalent operations or features defined for NVMe devices. The SNTL emulates a SCSI device server for all MODE SENSE commands and MODE SELECT commands, and shall emulate the mode pages listed in table 11.

The Mode Page Policy VPD page (see 11.5.7) should be implemented. If implemented, then the MODE PAGE POLICY field in each mode page policy descriptor should be set to 00b (i.e., shared) for each mode page and only one copy of mode page values should be maintained for all logical units within a target device (i.e., the MLUS bit is set to one in each mode page policy descriptor).

If the Mode Page Policy VPD page is not implemented, then the SNTL shall maintain shared mode pages for all I_T nexuses and shall share mode pages across all logical units within a target device.

11.4.2 Overview

The SNTL translations for mode pages are listed in table 11.

Table 11 — Summary of SCSI / ATA mode page mapping (part 1 of 2)

SCSI mode page	Reference
Command Duration Limit A (i.e., 0Ah/03h)	
Command Duration Limit B (i.e., 0Ah/04h)	
Control (i.e., 0Ah)	
Control Extension (i.e., 0Ah/01h)	
Read-Write Error Recovery (i.e., 01h)	
Caching (i.e., 08h)	
Informational Exceptions Control (i.e., 1Ch)	

Table 11 — Summary of SCSI / ATA mode page mapping (part 2 of 2)

SCSI mode page	Reference
Power Condition (i.e., 1Ah)	
Command Duration Limit T2A (i.e., 0Ah/07h)	
Command Duration Limit T2B (i.e., 0Ah/08h)	
Power Consumption (i.e, 1Ah/01h)	
All others	Unspecified

The format of mode parameter headers used for all pages is as described in 11.4.3. The format of the optional mode parameter block descriptors used for all mode pages is as described in 11.4.4.

The list of mode pages the SNTL may support are shown in table 11 and others that are defined in SPC-6, SBC-5, and ZBC-2 that do not have defined SNT translations.

The Command Duration Limit A mode page, the Command Duration Limit B mode page, the Command Duration Limit T2A mode page, and the Command Duration Limit T2B mode page may or may not be supported depending on the setting of the CDL_CTRL field xxx. If the setting specifies that any of these mode pages are not supported, then they shall not be returned in any parameter data that indicates supported mode pages and supported subpages (see SPC-6).

11.4.3 Mode parameter header

...

11.4.4 Mode parameter block descriptor fields

...

11.5 Vital product data parameters

11.5.1 Overview

The VPD page translations defined in this standard are described in table 12.

Table 12 – VPD page translations

VPD Page Name	Page code	Reference	Support
Block Device Characteristics	B1h	11.5.2	Optional
Block Limits	B0h	11.5.3	Optional
Device Identification	83h	11.5.4	Mandatory
Extended Inquiry Data	86h	11.5.5	Optional
Logical Block Provisioning	B2h	11.5.6	Optional
Mode Page Policy	87h	11.5.7	Optional
Supported Block Lengths and Protection Types	B4h	11.5.8	Optional
Supported VPD Pages	00h	11.5.9	Mandatory
Unit Serial Number	80h	11.5.10	Optional
All others	See applicable command standard		

11.5.2 Block Device Characteristics VPD page (i.e., B0h)**11.5.2.1 Block Device Characteristics VPD page (i.e., B0h) overview**

Table 13 shows the translation of fields in the Block Device Characteristics VPD page.

Table 13 – Block Device Characteristics VPD page field translations

Field	Description
PERIPHERAL QUALIFIER	Shall be set as described in 8.1.2
PERIPHERAL DEVICE TYPE	
PAGE CODE	Shall be set to B1h
PAGE LENGTH	Shall be set to 003Ch
MEDIUM ROTATION RATE	See 11.5.2.2.
PRODUCT TYPE	Shall be set to 00h (i.e., not indicated).
WABEREQ	Shall be set to 00b (i.e., not specified).
WACEREQ	Shall be set to 00b (i.e., not specified).
NOMINAL FORM FACTOR	Shall be set to 0h (i.e., nominal form factor is not reported).
MACT	Shall be set to 1b (i.e., multiple actuators) if: a) the SNTL supports Rotational Media; and b) the NUMA field in the Rotational Media Information Log Page (Log Page Identifier 16h) (see NVMeBASE) is greater than one; otherwise, shall be set to 0b (i.e., not multiple actuators).
RBWZ	Shall be set to 0b (i.e., REASSIGN BLOCKS command does not recover logical block data).
BOCS	Shall be set to 0b (i.e., background operational control is not supported).
FUAB	Shall be set to 1b (i.e., modern FUA and SYNCHRONIZE CACHE behavior)
VBULS	Shall be set to 00b (i.e., does not support verify byte check unmapped LBA).
DEPOPULATION TIME	Shall be set to zero (i.e., depopulation time is not reported).

11.5.2.2 MEDIUM ROTATION RATE field

The MEDIUM ROTATION RATE field shall be translated as defined in table 14.

Table 14 – MEDIUM ROTATION RATE field translation

SNTL supports Rotational Media	NVMe RMEDIA bit ^a	MEDIUM ROTATION RATE field
No	any	Shall be set to 0001h (i.e., non-rotating medium).
Yes	0	
	1	Shall be set to the value in the NRS field in the Rotational Media Information Log Page (Log Page Identifier 16h) (see NVMeBASE).
^a RMEDIA bit in the I/O Command Set Independent Identify Namespace Data Structure (CNS 08) NSFEAT field (see NVMeBASE)		

11.5.3 Block Limits VPD page (i.e., B0h)**11.5.3.1 Block Limits VPD page (i.e., B0h) overview**

Table 15 shows the translation of fields in the Block Limits VPD page.

Table 15 – Block Limits VPD page field translations

Field	Description
PERIPHERAL QUALIFIER	Shall be set as described in 8.1.2.
PERIPHERAL DEVICE TYPE	
PAGE CODE	Shall be set to B0h.
PAGE LENGTH	Shall be set to 003Ch.
WSNZ	Shall be set to 1b (i.e., WRITE SAME command does not support a zero number of blocks).
MAXIMUM COMPARE AND WRITE LENGTH	Shall be set to zero (i.e., COMPARE AND WRITE command not supported).
OPTIMAL TRANSFER LENGTH GRANULARITY	Shall be set to zero (i.e., not reported).
MAXIMUM TRANSFER LENGTH	Shall be set to zero (i.e., not reported).
OPTIMAL TRANSFER LENGTH	Shall be set to zero (i.e., not reported).
MAXIMUM PREFETCH LENGTH	Shall be set to zero (i.e., not reported).
MAXIMUM UNMAP LBA COUNT	See 11.5.3.2.

Table 15 – Block Limits VPD page field translations (Continued)

Field	Description
MAXIMUM UNMAP BLOCK DESCRIPTOR COUNT	<p>If DSM Supported (see 4.2.2) is:</p> <ul style="list-style-type: none"> a) false, then this field shall be set to zero; or b) true, then if the NVMe I/O Command Set Specific Identify Controller data structure (CNS 06h, CSI 00h) DMRL field (see NVMCS) is set to: <ul style="list-style-type: none"> A) zero, this field shall be set to a vendor specific value; or B) non-zero, this field shall be set to the minimum of: <ul style="list-style-type: none"> a) the value in the NVMe I/O Command Set Specific Identify Controller data structure (CNS 06h, CSI 00h) DMRL field (see NVMCS); or b) the maximum unmap block descriptor count supported by the SNTL.
OPTIMAL UNMAP GRANULARITY	Shall be set to zero (i.e., not reported).
UGAVALID	Shall be set to 0b (i.e., the UNMAP GRANULARITY ALIGNMENT field is not valid).
UNMAP GRANULARITY ALIGNMENT	Shall be set to zero (i.e., not reported).
MAXIMUM WRITE SAME LENGTH	See 11.5.3.3.
MAXIMUM ATOMIC TRANSFER LENGTH	Shall be set to zero (i.e., not reported).
ATOMIC ALIGNMENT	Shall be set to zero (i.e., not reported).
ATOMIC TRANSFER LENGTH GRANULARITY	Shall be set to zero (i.e., not reported).
MAXIMUM ATOMIC TRANSFER LENGTH WITH ATOMIC BOUNDARY	Shall be set to zero (i.e., not reported).
MAXIMUM ATOMIC BOUNDARY SIZE	Shall be set to zero (i.e., not reported).

11.5.3.2 MAXIMUM UNMAP LBA COUNT field

The MAXIMUM UNMAP LBA COUNT field shall be translated as defined in Table 16.

Table 16 – MAXIMUM UNMAP LBA COUNT field translation

DSM Supported (see 4.2.2)	NVMe NVMDSMSV bit ^a	NVMe DMSL field ^b	NVMe DMRSL field ^c	MAXIMUM UNMAP LBA COUNT field ^d
No	any	any	any	Shall be set to zero
Yes	1	non-zero	non-zero	Shall be set to MIN(FFFF_FFFFh, VS_ ULC _MAX)
	any	zero	zero	
	0	non-zero	non-zero	Shall be set to MIN(DMSL, DMRSL, VS_ ULC _MAX)
	any	zero	non-zero	Shall be set to zero ^e
	any	non-zero	zero	
^a NVMDSMSV bit in the NVMe Identify Controller data structure (CNS 01h) ONCS field (see NVMeBASE) (0b = Absolute maximum, 1b = Recommended Maximum) ^b NVMe I/O Command Set Specific Identify Controller data structure (CNS 06h, CSI 00h) DMSL field (see NVMCS) ^c NVMe I/O Command Set Specific Identify Controller data structure (CNS 06h, CSI 00h) DMRSL field (see NVMCS) ^d VS_ ULC _MAX = The maximum unmap lba count supported by the SNTL. ^e These combinations are prohibited by NVMe 2.0				

11.5.3.3 MAXIMUM WRITE SAME LENGTH field

The MAXIMUM WRITE SAME LENGTH field shall be translated as defined in table 17.

Table 17 – MAXIMUM WRITE SAME LENGTH field translation

Logical Block Length ^c	NVMe NVMWZSV bit ^a	NVMe WZSL field ^b	Interleaved Protection Information	MAXIMUM WRITE SAME LENGTH field ^{b, d, e}
512	0	non-zero	false	Shall be set to MIN (((2 ^ (WZSL +12)) / 512), VS_WSL_MAX_512)
512	0	non-zero	true	Shall be set to MIN (((2 ^ (WZSL+12)) / 520), VS_WSL_MAX_512)
512	0	0	any	Shall be set to VS_WSL_MAX_512
512	1	any	any	
4096	0	non-zero	false	Shall be set to MIN (((2 ^ (WZSL+12)) / 4096), VS_WSL_MAX_4K)
4096	0	non-zero	true	Shall be set to MIN (((2 ^ (WZSL+12)) / 4104), VS_WSL_MAX_4K)
4096	0	0	any	Shall be set to VS_WSL_MAX_4K
4096	1	any	any	
^a nvmwzsv bit in the NVMe Identify Controller data structure (CNS 01h) ONCS field (see NVMeBASE) (0b = Absolute maximum, 1b = Recommended Maximum)				
^b WZSL = value in NVMe I/O Command Set Specific Identify Controller data structure (CNS 06h, CSI 00h) WZSL field (see NVMCS)				
^c See 5.4.				
^d VS_WSL_MAX_512 = The maximum write same length supported by the SNTL for 512 byte Logical Block Length (e.g., 32,768 (16MiB effective))				
^e VS_WSL_MAX_4K = The maximum write same length supported by the SNTL for 4096 byte Logical Block Length (e.g., 8,192 (32MiB effective))				

11.5.4 Device Identification VPD page

...

11.5.5 Extended INQUIRY Data VPD page (i.e., 86h)

Table 18 shows the translation of fields in the Extended INQUIRY Data VPD page.

Table 18 – Extended INQUIRY Data VPD page field translations (part 1 of 3)

Field	Description
PERIPHERAL QUALIFIER	Shall be set as described in 8.1.2
PERIPHERAL DEVICE TYPE	
PAGE CODE	Shall be set to 86h.
PAGE LENGTH	Shall be set to 003Ch.
ACTIVATE MICROCODE	Shall be set to 00b (i.e., actions of the device server are not indicated).
SPT	Shall be set to 000b (i.e., only type 1 supported) if: a) the SNTL does not support protection information; or b) the PIT1S bit, the PIT2S bit, and the PIT3S bit in the NVMe Identify Namespace data structure (CNS 00h) DPC field (see NVMCS) are set to 0b; otherwise this field shall be set to 110b (i.e., see the Supported Block Lengths And Protection Types VPD page).
GRD_CHK	Shall be set to 0b if: a) the SNTL does not support protection information; or b) the PIT field in the NVMe Identify Namespace data structure (CNS 00h) DPS field (see NVMCS) is set to 000b (i.e., protection information disabled), otherwise this bit shall be set to 1b.
APP_CHK	Shall be set to 0b if: a) the SNTL does not support protection information; or b) the PIT field in the NVMe Identify Namespace data structure (CNS 00h) DPS field (see NVMCS) is set to 000b (i.e., protection information disabled), otherwise this bit shall be set to 1b.
REF_CHK	Shall be set to 0b if: a) the SNTL does not support protection information; or b) the PIT field in the NVMe Identify Namespace data structure (CNS 00h) DPS field (see NVMCS) is set to 000b (i.e., protection information disabled), otherwise this bit shall be set to 1b.
UASK_SUP	Shall be set to 1b (i.e., unit attention sense key specific data supported).
GROUP_SUP	Shall be set to 0b (i.e., grouping function not supported).
PRIOR_SUP	Shall be set to 0b (i.e., command priority is not supported).
HEADSUP	Shall be set to 0b (i.e., HEAD OF QUEUE task attribute not supported).
ORDSUP	Shall be set to 0b (i.e., ORDERED task attribute not supported).
SIMPSUP	Shall be set to 1b (i.e., SIMPLE task attributed supported).
WU_SUP	Shall be set to the value of the NVMWUSV bit in the NVMe Identify Controller data structure (CNS 01h) ONCS field (see NVMeBASE).
NV_SUP	Shall be set to 0b (i.e., may or may not support nonvolatile cache).

Table 18 – Extended INQUIRY Data VPD page field translations (part 2 of 3)

Field	Description
V_SUP	Shall be set to the value of the VWCP bit in the NVMe Identify Controller data structure (CNS 01h) vwc field (see NVMeBASE).
NO_PI_CHK	Shall be set to 0b (i.e., protection information checks as specified by the WRPROTECT bit).
P_I_I_SUP	Shall be set to 0b (i.e., protection information intervals not supported).
LUICLR	Shall be set to 1b (i.e., modern logical unit I_T nexus clear behavior).
LU COLLECTION TYPE	Shall be set to 0b (i.e., not reported).
R_SUP	Shall be set to 0b (i.e., not supported).
RTD_SUP	Shall be set to 0b (i.e., RTD bit not supported).
HSSRELEF	Shall be set to 0b (i.e., normal reset handling).
MULTI I_T NEXUS MICROCODE DOWNLOAD	Shall be set to 0h (i.e., vendor specific).
EXTENDED SELF-TEST COMPLETION MINUTES	Shall be set to the NVMe Identify Controller data structure (CNS 01h) EDSTT field.
POA_SUP	Shall be set to 0b (i.e., not supported).
HRA_SUP	Shall be set to 0b (i.e., not supported).
VSA_SUP	Shall be set to 0b (i.e., not supported).
DMS_VALID	Shall be set to 1b (i.e., valid).
TPSBV	Shall be set to 0b (i.e., support for individual time policies is not indicated).
MAXIMUM SUPPORTED SENSE DATA LENGTH	Shall be set to zero (i.e., not reported).
IBS	Shall be set to 0b (i.e., not supported).
IAS	Shall be set to 0b (i.e., not supported).
SAC	Shall be set to 0b (i.e., not supported).
NRD1	Shall be set to 0b (i.e., not supported).
NRD0	Shall be set to 0b (i.e., not supported).
MAXIMUM INQUIRY CHANGE LOGS	Shall be set to zero.
MAXIMUM MODE PAGE CHANGE LOGS	Shall be set to zero.
DM_MD_4	Shall be set to 0b (i.e., not supported).
DM_MD_5	Shall be set to 0b (i.e., not supported).
DM_MD_6	Shall be set to 0b (i.e., not supported).
DM_MD_7	Shall be set to 0b (i.e., not supported).
DM_MD_D	Shall be set to 0b (i.e., not supported).
DM_MD_E	Shall be set to 1b (i.e., supported).
DM_MD_F	Shall be set to 1b (i.e., supported).
Inactive time policies supported descriptor	Shall be set to zero (i.e., no policies supported).

Table 18 – Extended INQUIRY Data VPD page field translations (part 3 of 3)

Field	Description
Active time policies supported descriptor	Shall be set to zero (i.e., no policies supported).
Total time policies supported descriptor	Shall be set to zero (i.e., no policies supported).

11.5.6 Logical Block Provisioning VPD page

Table 19 shows the translation of fields in the Logical Block Provisioning VPD page.

Table 19 – Logical Block Provisioning VPD page field translations

Field	Description
PERIPHERAL QUALIFIER	Shall be set as described in 8.1.2.
PERIPHERAL DEVICE TYPE	
PAGE CODE	Shall be set to B2h.
PAGE LENGTH	Shall be set to 0004.
THRESHOLD EXPONENT	Shall be set to 00h (i.e., threshold sets not supported).
LBPV	If DSM Supported (see 4.2.2) is true, then the SNTL shall set this bit to 1b (i.e., UNMAP command supported); otherwise, the SNTL shall set this bit to 0b.
LBPWS	If: a) WZ Supported (see 4.2.3) is true; and b) the WZDS bit in the NVMe Identify Namespace data structure DLFEAT field (see NVMCS) is set to 1b (i.e., Write Zeroes Deallocate bit supported), then the SNTL shall set this bit to 1b (i.e. WRITE SAME (16) command is able to unmap LBAs); otherwise, the SNTL shall set this bit to 0b.
LBPWS10	If: a) WZ Supported (see 4.2.3) is true; and b) the WZDS bit in the NVMe Identify Namespace data structure DLFEAT field (see NVMCS) is set to 1b (i.e., Write Zeroes Deallocate bit supported), then the SNTL shall set this bit to 1b (i.e., WRITE SAME (10) command is able to unmap LBAs); otherwise, the SNTL shall set this bit to 0b.
LBPRZ	If the DRB field in the NVMe Identify Namespace data structure DLFEAT field (see NVMCS) is set to 001b (i.e., deallocated logical block returns zeros), then this field shall be set to 001b (i.e., returns zeros), otherwise this field shall be set to 000b (i.e., vendor specific).
ANC_SUP	Shall be set to 0b (i.e., not supported).
DP	Shall be set to 0b (i.e., PROVISIONING GROUP DESCRIPTOR field not reported).
MINIMUM PERCENTAGE	Shall be set to 00000b (i.e., not reported).
PROVISIONING TYPE	Shall be set to 001b (i.e., resource provisioned).
THRESHOLD PERCENTAGE	Shall be set to 00h (i.e., not supported).
PROVISIONING GROUP DESCRIPTOR	This field is not returned.

11.5.7 Mode Page Policy VPD page (i.e., B7h)**11.5.7.1 Mode Page Policy VPD page (i.e., B7h) overview**

Table 20 shows the translation of fields in the Mode Page Policy VPD page.

Table 20 – Mode Page Policy VPD page field translations

Field	Description
PERIPHERAL QUALIFIER	Shall be set as described in 8.1.2.
PERIPHERAL DEVICE TYPE	
PAGE CODE	Shall be set to 87h.
PAGE LENGTH	Shall be set to 04h.
Mode page policy descriptor list	The SNTL shall support one Mode page policy descriptor (see 11.5.7.2).

11.5.7.2 Mode Page Policy Descriptor translation

Table 15 shows the translation for the fields of a Mode page policy descriptor.

Table 21 – Mode Page Policy Descriptor field translations

Field	Description
POLICY PAGE CODE	Shall be set to 3Fh (i.e., all mode pages).
POLICY SUBPAGE CODE	Shall be set to FFh (i.e., all subpages).
MLUS	Shall be set to 1b (i.e., shared across all logical units).
MODE PAGE POLICY	Shall be set to 01b (i.e., per target port).

11.5.8 Supported Block Lengths and Protection Types VPD page**11.5.8.1 Supported Block Lengths and Protection Types VPD page overview**

Table 22 shows the translation of fields in the Supported Block Lengths and Protection Types VPD page. If the SNTL supports protection information, then this page shall be supported.

Table 22 – Supported Block Lengths and Protection Types VPD page field translations

Field	Description
PERIPHERAL QUALIFIER	Shall be set as described in 8.1.2.
PERIPHERAL DEVICE TYPE	
PAGE CODE	Shall be set to B4h.
PAGE LENGTH	Shall be set to the number of bytes in the Logical block length and protection types descriptor list.
Logical block length and protection types descriptor list.	See 11.5.8.2.

11.5.8.2 Logical block length and protection types descriptor list

[Editor's Note 11: Review this translation once spec is more complete](#)

The SNTL shall return a Logical block length and protection types descriptor for each format that is supported by both the SNTL and the NVMe Controller as reported in the NVMe Identify Namespace data structure (CNS 00h) LBAF0 field through LBAF63 fields (see NVMCS).

If the SNTL supports protection information, then two descriptors need to be reported for a given NVMe LBA format. The first descriptor has the T0PS bit set to 1b (i.e., type 0 protection supported). The second descriptor has one or more of the T1PS bit, the T2PS bit, or the T3PS bit set.

Table 23 shows the translation of a Logical block length and protection types descriptor.

Table 23 – Logical Block Length and Protection Types descriptor translation

Field	Description
LOGICAL BLOCK LENGTH	Shall be set to $(1 \ll \text{LBADS field})$.
P_I_I_SUP	Shall be set to 0b (i.e., protection information interval supported).
NO_PI_CHK	Shall be set to 1b (i.e., protection information checking disabled).
GRD_CHK	Shall be set to 1b (i.e., LOGICAL BLOCK GUARD field checked).
APP_CHK	Shall be set to 1b (i.e., LOGICAL BLOCK APPLICATION TAG field checked).
REF_CHK	Shall be set to 1b (i.e., LOGICAL BLOCK REFERENCE TAG field checked).
T3PS	Shall be set to the value of the PIT3S bit of the NVMe Identify Namespace data structure (CNS 00h) DPC field (see NVMCS), if this descriptor supports protection information, otherwise this bit shall be set to 0.
T2PS	Shall be set to the value of the PIT2S of the NVMe Identify Namespace data structure (CNS 00h) DPC field (see NVMCS), if this descriptor supports protection information, otherwise this bit shall be set to 0.
T1PS	Shall be set to the value of the PIT1S bit of the NVMe Identify Namespace data structure (CNS 00h) DPC field (see NVMCS), if this descriptor supports protection information, otherwise this bit shall be set to 0b.
T0PS	Shall be set to 1b (i.e., supported) if this descriptor does not support protection information, otherwise this bit shall be set to 0b.

Editor's Note 12: As I recall, the example text should be indented?

EXAMPLE - An NVMe Controller that returns four LBAF fields along with the resulting descriptors are shown in Table 24.

Table 24 – Descriptor Example

NVMe LBAF Fields		SCSI descriptors		
LBADS	MS	LOGICAL BLOCK LENGTH field	T0PS bit	T1PS, T2PS, and/or T3PS bits
9 (512)	0	512	1b	0b
9 (512)	8	512	1b	0b
		512	0b	1b
12 (4K)	0	4096	1b	0b
12 (4K)	8	4096	1b	0b
		4096	0b	1b
12 (4K)	16	If not supported by SNTL, then no descriptors reported.		

11.5.9 Supported VPD Pages VPD page

Table 25 shows translation of fields in the Supported VPD Pages VPD page.

Table 25 – Supported VPD Pages VPD page field translations

Field	Description
PERIPHERAL QUALIFIER	Shall be set as described in 8.1.2.
PERIPHERAL DEVICE TYPE	
PAGE CODE	Shall be set to 00h.
PAGE LENGTH	Shall be set to the number of bytes in the Supported VPD page list.
Supported VPD page list	Shall contain the list of supported VPD page codes in ascending order beginning with page code 00h and should include all page codes indicated by the supported VPD page translations defined in Table 12.

11.5.10 Unit Serial Number VPD page

...

12 Translation of NVMe errors to SCSI errors

12.1 Overview

Unless otherwise specified in the subclause describing the translation of a particular SCSI command, log page, mode page, or VPD page, the SNTL shall translate ATA commands that complete with an error to SCSI errors as shown in table xxx.

13 SNT specific SCSI extensions

13.1 Overview

This clause defines additional SCSI commands, mode pages, security protocols, and VPD pages that may be supported by a SNTL to provide capabilities in addition to the capabilities defined in the other SCSI command sets.

For SCSI commands specific to SNTL implementations see 13.2.

For Mode pages specific to SNTL implementations see 13.3.

For VPD pages specific to SNTL implementations see 13.4.

For Security protocols specific to SNTL implementations see 13.5.

For log pages specific to SNTL implementations see 13.6.

13.2 SNT specific SCSI extensions

[Editor's Note 13: NVMe pass through goes here](#)

13.3 SNT specific mode pages

13.4 SNT specific VPD pages

13.5 SNT specific security protocols

13.6 SNT specific log pages

Annex A
(informative)

Sample algorithms for splitting commands

A.1 TBD

Bibliography

T10/BSR INCITS 465 *SCSI/ATA Translation-2 (SAT-2)* (planned as ISO/IEC 14776-922)

Memory Stick™ (MS). One Stop Site for Formats

NOTE 2 - Available from <https://www.oss-formats.org>.

MultiMediaCard (e•MMC), JEDEC®

NOTE 3 - Available from <https://www.jedec.org>.

NOTE 4 - JEDEC® is a registered trademark of JEDEC Solid State Technology Association. This information is given for the convenience of users of this standard and does not constitute an endorsement by ISO or IEC.