SPDM to Storage Binding Specification

Version: 1.0.0

Document Identifier: DSP0286

Date: 2025-05-15

Version History: https://www.dmtf.org/dsp/DSP0286

Supersedes: None

Document Class: Normative

Document Status: Published

Document Language: en-US

10 DMTF is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. Members and non-members may reproduce DMTF specifications and documents for uses consistent with this purpose, provided that correct attribution is given. As DMTF specifications may be revised from time to time, the particular version and release date should always be noted.

11 Implementation of certain elements of this standard or proposed standard may be subject to third-party patent rights, including provisional patent rights (herein "patent rights"). DMTF makes no representations to users of the standard as to the existence of such rights and is not responsible to recognize, disclose, or identify any or all such third-party patent right owners or claimants, nor for any incomplete or inaccurate identification or disclosure of such rights, owners, or claimants. DMTF shall have no liability to any party, in any manner or circumstance, under any legal theory whatsoever, for failure to recognize, disclose, or identify any such third-party patent rights, or for such party's reliance on the standard or incorporation thereof in its products, protocols, or testing procedures. DMTF shall have no liability to any party implementing such standards, whether such implementation is foreseeable or not, nor to any patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a patent owner for such implementations.

12 For information about patents held by third-parties which have notified DMTF that, in their opinion, such patents may relate to or impact implementations of DMTF standards, visit https://www.dmtf.org/about/policies/disclosures.

13 The NVM Express®, NVMe®, and NVMe-oF™ word marks are registered and unregistered trademarks and service marks of NVM Express, Inc. in the United States and other countries. Unauthorized use strictly prohibited. PCI-SIG® and PCIe® are registered trademarks or service marks of PCI-SIG. CompactFlash® is a registered trademark of Sandisk LLC. CFast® is a registered trademark of CompactFlash Association. All other marks and brands are the property of their respective owners.

14 This document's normative language is English. Translation into other languages is permitted.

CONTENTS

1 Foreword

The Security Protocols and Data Models (SPDM) Working Group prepared the *SPDM to Storage Binding Specification* (DSP0286).

DMTF is a not-for-profit association of industry members that promotes enterprise and systems management and interoperability. For information about DMTF, see https://www.dmtf.org.

1.1 Acknowledgments

DMTF acknowledges the following individuals for their contributions to this document:

- Jeff Anderson — Google
- Steven Bellock — NVIDIA Corporation
- James Borden — Kioxia
- Daniil Egranov — Arm Limited
- Jim Hatfield — Seagate Technology
- Philip Hawkes — Qualcomm Inc.
- Brett Henning — Broadcom Inc.
- Jeff Hilland — Hewlett Packard Enterprise
- Guerney Hunt — IBM
- Raghupathy Krishnamurthy — NVIDIA Corporation
- Eliel Louzoun — Intel Corporation
- Wilfred Mallawa — Western Digital Technologies, Inc.
- Chandra Nelogal — Dell Technologies
- Alexander Novitskiy — Intel Corporation
- Scott Phuong — Microsoft Corporation
- Xiaoyu Ruan — Intel Corporation
- Yoni Shternhell — Western Digital Technologies, Inc.
- Jiewen Yao — Intel Corporation
- Wilson Young — Solidigm

2 Introduction

SPDM to Storage Binding Specification binds SPDM messages (DSP0274) and SPDM Secured Messages (DSP0277) to storage transports and defines messages and data objects to enable this binding. This binding specification enables the extension of the capabilities defined in the *Security Protocol and Data Model (SPDM) Specification* to storage devices. Further, this binding specification enables the use of intermediate devices, such as a host bus adapter, between the Requester and the storage device.

This specification supports the following storage interfaces:

- NVM Express® (NVMe®), including NVMe-oF™
- Small Computer System Interface (SCSI)
- Advanced Technology Attachment (ATA)

2.1 Document conventions

- Document titles appear in *italics*.
- The first occurrence of each important term appears in *italics* with a link to its definition.
- ABNF rules appear in a monospaced font.

2.2 Reserved and unassigned values

Unless otherwise specified, any reserved, unspecified, or unassigned values in enumerations or other numeric ranges are reserved for future definition by DMTF.

Unless otherwise specified, field values marked as Reserved shall be written as zero (`0`), ignored when read, not modified, and not interpreted as an error if not zero, and used in transcript hash calculations as is.

2.3 Byte ordering

Unless otherwise specified, byte ordering of multi-byte numeric fields or multi-byte bit fields is *little endian* (that is, the lowest byte offset holds the least significant byte, and higher offsets hold the more significant bytes).

For protocol-specific byte ordering details, see the following subclauses:

- NVMe byte ordering
- SCSI byte ordering
- ATA byte ordering

2.4 Other conventions

Unless otherwise specified, all figures are informative.

3 Scope

This document defines the format of Security Protocol and Data Model (SPDM) messages over storage protocols.

3.1 Out of scope

The following topics are out of scope for this specification:

- Asynchronous notification from the Responder to the Requester
- Translation of commands between different storage protocols
- Interactions with other interface commands
- Interference with the use of any other security protocols

3.2 Normative references

The following referenced documents are indispensable for the application of this specification. For dated or versioned references, only the edition cited (including any corrigenda or DMTF update versions) applies. For references without a date or version, the latest published edition of the referenced document (including any corrigenda or DMTF update versions) applies.

- ACS-5, ATA Command Set — 5, ISO/IEC 17760-105:2024, https://www.iso.org/standard/86635.html
- DMTF DSP0274, *Security Protocol and Data Model (SPDM) Specification*, https://www.dmtf.org/dsp/DSP0274
- DMTF DSP0277, *Secured Messages using SPDM Specification*, https://www.dmtf.org/dsp/DSP0277
- *ISO/IEC Directives, Part 2, Principles and rules for the structure and drafting of ISO and IEC documents — 2021 (9th edition)*, https://www.iso.org/sites/directives/current/part2/index.xhtml
- IETF RFC 5234, *Augmented BNF for Syntax Specifications: ABNF*, January 2008, https://tools.ietf.org/html/rfc5234
- NVM Express, *NVM Express® Base Specification* Revision 2.1 (August 5th, 2024), https://web.archive.org/web/20250308094154if_/https://nvmexpress.org/wp-content/uploads/NVM-Express-Base-Specification-Revision-2.1-2024.08.05-Ratified.pdf
- SAM-6, SCSI Architectural Model — 6, ANSI INCITS 546-2021, https://webstore.ansi.org/standards/incits/incits5462021
- SBC-5, SCSI Block Commands — 5, ANSI INCITS 571, https://www.t10.org/cgi-bin/ac.pl?t=f&f=sbc5r08.pdf
- SPC-6, SCSI Primary Commands — 6, ANSI INCITS 566, https://www.t10.org/cgi-bin/ac.pl?t=f&f=spc6r13.pdf
- *The Datagram Transport Layer Security (DTLS) Protocol Version 1.3* (RFC 9147), April 2022, https://datatracker.ietf.org/doc/rfc9147/

3.3 Terms and definitions

In this document, some terms have a specific meaning beyond the normal English meaning. This clause defines those terms.

The terms "shall" ("required"), "shall not," "should"("recommended"), "should not" ("not recommended"), "may," "need not" ("not required"), "can" and "cannot" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 7. The terms in parentheses are alternatives for the preceding term, for use in exceptional cases when the preceding term cannot be used for linguistic reasons. Note that ISO/IEC Directives, Part 2, Clause 7 specifies additional alternatives. Occurrences of such additional alternatives shall be interpreted in their normal English meaning.

The terms "clause," "subclause," "paragraph," and "annex" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 6.

The terms "normative" and "informative" in this document are to be interpreted as described in ISO/IEC Directives, Part 2, Clause 3. In this document, clauses, subclauses, or annexes labeled "(informative)" do not contain normative content. Notes and examples are always informative elements.

The terms that DSP0274 defines also apply to this document.

This specification uses these terms:

Term	Definition
`IF-SEND`	Generic term for a security-related command that transfers data from the initiator to the target. For NVMe, this is Security Send. For SCSI, this is SECURITY PROTOCOL OUT. For ATA, this is TRUSTED SEND or TRUSTED SEND DMA.
`IF-RECV`	Generic term for a security-related command that transfers data from the target to the initiator. For NVMe, this is Security Receive. For SCSI, this is SECURITY PROTOCOL IN. For ATA, this is TRUSTED RECEIVE or TRUSTED RECEIVE DMA.

3.3.1 Equivalent terms

This binding specification primarily uses SPDM terminology. The following table explains how these terms align with terms from the underlying storage protocol specifications.

SPDM Binding Specification Term	SCSI Term	ATA Term	NVMe Term
`IF-SEND`	SECURITY PROTOCOL OUT	TRUSTED SEND or TRUSTED SEND DMA	Security Send
`IF-RECV`	SECURITY PROTOCOL IN	TRUSTED RECEIVE or TRUSTED RECEIVE DMA	Security Receive

SPDM Binding Specification Term	SCSI Term	ATA Term	NVMe Term
Requester	SCSI initiator	Host	Host
Responder	SCSI target device	Device	Controller
Storage protocol	Service delivery subsystem	SATA protocol	NVMe protocol

89 ## 3.4 Symbols and abbreviated terms

90 The abbreviations and notations defined in DSP0274 apply to this document.

91 ## 3.5 Binding Information

92 This version of this specification binds to **all** published versions of the *Security Protocol and Data Model (SPDM) Specification* (DSP0274), though some functionality might not be available under all versions.

93 This version of this specification binds to these versions of the *Secured Messages using SPDM Specification* (DSP0277):

94 • Version 1.0.0 and all 1.0 errata versions
95 • Version 1.1.0 and all 1.1 errata versions
96 • Version 1.2.0 and all 1.2 errata versions

97 ## 3.6 Annotation of differences between storage protocols

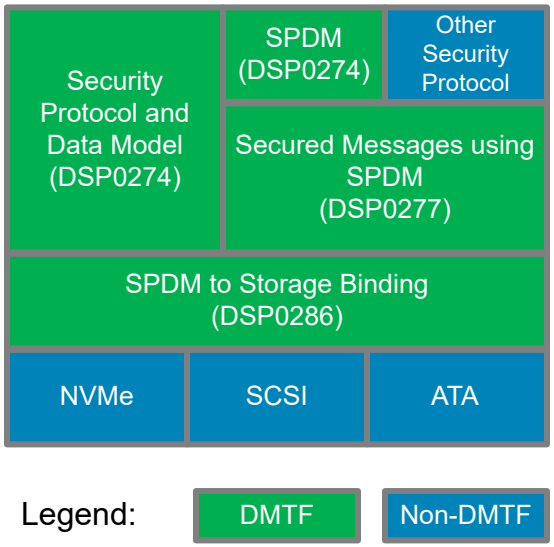
98 Where differences are not noted, this specification applies to all storage protocols that are in scope.

99

4 Theory of operation

100 **Figure 1 — SPDM storage stack**

101



102 Figure 1 — SPDM storage stack shows a high-level view of the SPDM to Storage Binding. The SPDM to Storage Binding allows endpoints to send messages defined by the *Security Protocol and Data Model (SPDM) Specification* (DSP0274) between storage endpoints. Further, this specification allows endpoints to exchange messages using the *Secured Messages using SPDM Specification* (DSP0277). SPDM Secured Messages encapsulate other messages. As shown, SPDM Secured Messages can encapsulate SPDM messages and other `IF-SEND` and `IF-RECV` commands.

103 There are important differences between SPDM operating on a storage interface and SPDM operating on an MCTP interface. These differences are outlined in Table 1 — Differences between Storage and MCTP Interfaces.

104 **Table 1 — Differences between Storage and MCTP Interfaces**

Category	MCTP Behavior	SPDM over Storage Behavior
Which device issues a request	Either endpoint	Only the initiator
Which device responds	Either endpoint	Only the target
Bit and byte order (endianness)	SPDM is little endian, while MCTP is big endian.	Varies by storage interface
Request and response timing	There is a specification-defined time limit between issuing a request and receiving a response.	There is no interface-defined limit between the request and response because they are separate interface-level commands. SPDM-defined timing parameters shall be the minimum time allowed to process a request.

5 SPDM Storage commands

The following commands are defined by this specification to manage the SPDM characteristics of the storage protocol.

5.1 Security protocol commands

SPC-6 reserves the use of SECURITY PROTOCOL Code `0xE8` for `IF-SEND` and `IF-RECV` commands for use by DMTF. This document specifies the use of this code with SPDM specifications. The details of the `IF-SEND` and `IF-RECV` commands are found in the following clauses:

- NVMe command format
- SCSI command format
- ATA command format

The `ConnectionID` field shall identify the SPDM connection for a command, as DSP0274 describes. All devices shall support `ConnectionID` 0 and shall support `ConnectionID` s from 0 to the value of the `MaxConnectionID` field, contiguously, in the SPDM Storage Discovery response.

The `SPDMOperation` field shall identify the `SPDM Storage Operation Code` , as Table 2 — SPDM Storage Operation Codes defines.

Some `SPDMOperation` s only support `IF-SEND` or `IF-RECV` , as Table 2 — SPDM Storage Operation Codes defines.

Table 2 — SPDM Storage Operation Codes

`SPDMOperation` Value	Command	Mandatory	`IF-SEND` Support	`IF-RECV` Support	Description
`0x01`	SPDM Storage Discovery	Mandatory	No	Yes	See SPDM Storage Discovery.
`0x02`	SPDM Storage Pending Info	Optional	No	Yes	See SPDM Storage Pending Info.
`0x05`	SPDM Storage Message	Mandatory	Yes	Yes	See SPDM Storage Message.
`0x06`	SPDM Storage Secured Message	Optional	Yes	Yes	See SPDM Storage Secured Message.
All other values	Reserved				Reserved.

116 ## 5.2 SPDM Storage Response Header

117 The `SPDM Storage Response Header` is a common header structure that is used for SPDM Storage command responses. The `SPDM Storage Response Header` shall use the definition in Table 3 — SPDM Storage Response Header.

118 **Table 3 — SPDM Storage Response Header**

Byte offset	Field	Size (bytes)	Description
0	`DataLength`	2	The `DataLength` field shall return the number of available bytes in the SPDM Storage Data response buffer, not including any pad fields. This value might exceed the Allocation Length in the `IF-RECV` command, which indicates that the Responder has additional data that was not returned because the response buffer was not large enough.
2	`StorageBindingVersion`	2	The `StorageBindingVersion` for devices that implement this version of the SPDM to Storage Binding Specification shall be set to `0x1000`. Table 4 — Storage binding version format defines the format of this field.

119 Table 4 — Storage binding version format shall define the format of the `StorageBindingVersion` field.

120 **Table 4 — Storage binding version format**

Bit	Field	Value
[15:12]	`MajorVersion`	Shall be the major version of the storage protocol binding. See DSP0274 for description of major version.
[11:8]	`MinorVersion`	Shall be the minor version of the storage protocol binding. See DSP0274 for description of minor version.

Bit	Field	Value
[7:4]	`UpdateVersionNumber`	Shall be the update version of the storage protocol binding. See DSP0274 for description of update version.
[3:0]	`Alpha`	Shall be the alpha version of the storage protocol binding. For released versions, this field shall be zero. See DSP0274 for description of alpha version.

5.3 SPDM Storage Discovery

The `SPDM Storage Discovery` command reads the SPDM parameters from a device and is read using `IF-RECV`. The `SPDMOperation` field shall use the value specified in Table 2 — SPDM Storage Operation Codes for `SPDM Storage Discovery`. The format of the existing definition of the `SPDM Storage Discovery` response data is consistent across versions of this binding specification. The length of the response data might grow in future revisions, and reserved fields might gain new meanings.

The length of the `SPDM Storage Discovery` response data for this version of the *SPDM to Storage Binding Specification* shall be 32 bytes.

Table 5 — SPDM Storage Discovery response data

Byte offset	Field	Size (bytes)	Description
0	`StorageResponseHeader`	4	The `SPDM Storage Response Header` as SPDM Storage Response Header describes
4	`ConnParams`	1	Connection Parameters. Bit [1:0]. MaxConnectionID. The `MaxConnectionID` field shall contain the maximum `ConnectionID` for the device. A value of 0 indicates that the device supports one connection. The `MaxConnectionID` and `ConnectionID` fields are specific to the device and storage protocol in use and should not apply to other protocols. Bit [7:2]. Reserved.
5	Reserved	3	Reserved.

Byte offset	Field	Size (bytes)	Description
8	`SupportedOperations`	1	The `SupportedOperations` field shall return a bit mask of the `SPDMOperation`s that the Responder supports. The bit position corresponding to an `SPDMOperation` enumeration shall be set to `1` to indicate support for the corresponding `SPDMOperation`, for both Mandatory and Optional `SPDMOperation`s. For example, if a Responder supports `SPDM Storage Message`, it would set Bit[5] of `SupportedOperations` to `1`.
9	Reserved	7	Reserved for `SupportedOperations`.
16	Reserved	16	Reserved.

125 ## 5.4 SPDM Storage Pending Info

126 The `SPDM Storage Pending Info` command returns information about pending response data being held by the endpoint for the connection indicated in the requested `ConnectionID` and is read using `IF-RECV`. The `SPDM Storage Pending Info` command shall only be used with `SPDM Storage Message` and `SPDM Storage Secured Message`.

127 In storage protocols, the Requester must allocate a buffer to be used with the `IF-RECV` command but the Requester does not know the size of the response data that the Responder has pending. The Requester can use one or more of the following approaches to manage the size of the `IF-RECV` buffer for response data.

128 • Use a lookup table or other similar mechanism to predict the size of the response data.

129 • If both endpoints support SPDM Large Messages (`CHUNK_CAP` = `1`), the Requester can allocate a receive buffer of at least `DataTransferSize`.

130 • If the Responder supports the `SPDM Storage Pending Info` message, the Requester can use the response data to ensure that it allocates a large enough buffer.

131 The format of the `SPDM Storage Pending Info` response data is fixed for a given version of the *SPDM to Storage Binding Specification*, so the Requester can reliably know the expected length of the response data.

132 If a Responder uses the SPDM Large Message transfer mechanism to break a response into chunks, the `SPDM Storage Pending Info` response data does not update during the Large Message transfer. When the currently pending response has completed transmission, the Responder shall clear the `SPDM Storage Pending Info` response `ValidResponse` flag until a new command is received.

133 The `SPDMOperation` field shall use the value specified in Table 2 — SPDM Storage Operation Codes for `SPDM Storage Pending Info`.

134 The length of the `SPDM Storage Pending Info` response data for this version of the *SPDM to Storage Binding Specification* shall be 12 bytes.

135 **Table 6 — SPDM Storage Pending Info response data**

Byte offset	Field	Size (bytes)	Description
0	`StorageResponseHeader`	4	The `SPDM Storage Response Header` as SPDM Storage Response Header describes
4	`PendingInfoFlags`	4	The `PendingInfoFlags` field shall contain flags regarding the `SPDM Storage Pending Info` response. • Bit[0]. ValidResponse. Shall be set to `1` to indicate that a pending response is available and the `ResponseLength` field contains a valid pending response length. This bit shall be set to `0` to indicate that a valid response is not available. All other values reserved.
8	`ResponseLength`	4	The `ResponseLength` field shall return the number of available bytes in the currently pending response message for the connection indicated in the `ConnectionID` field. The value in this field shall be set to a non-zero value if the `PendingInfoFlags`.`ValidResponse` bit is set to `1`. Otherwise, the value in this field shall be set to `0`.

136

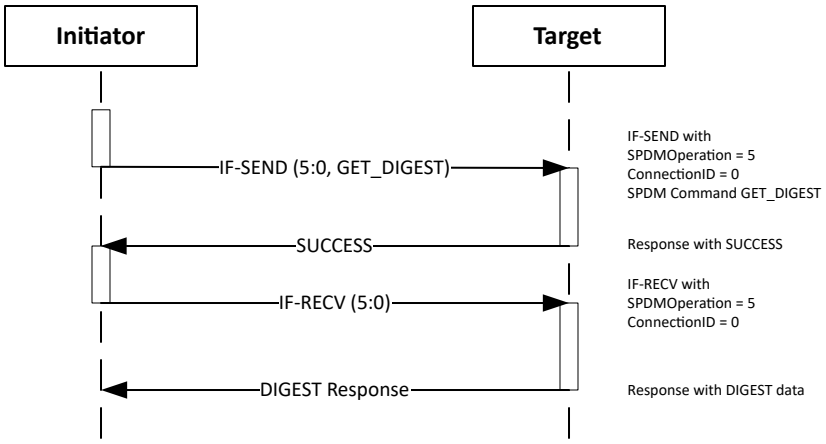
137

5.5 SPDM Storage Message

138 The `SPDM Storage Message` uses `IF-SEND` and `IF-RECV` to send and receive SPDM messages, as defined in DSP0274, between an initiator and a target. This binding specification requires that `SPDM Storage Message`s be used in a request/response flow and does not support targets sending messages asynchronously.

139 As Figure 2 — Storage message flow shows, `SPDM Storage Message` commands are unidirectional. An individual command can only send data or receive data. To handle this restriction, the Storage Binding Specification uses the `IF-SEND` and `IF-RECV` commands in a pair. An `IF-SEND` is used to send a request and any associated data from the initiator to the target. The target completes this command with a protocol-level status. The initiator then sends an `IF-RECV` from the initiator to the target. Finally, the target sends the SPDM response message and protocol-level status to complete the `IF-RECV`.

140 **Figure 2 — Storage message flow**



141 If a Responder receives an unexpected `IF-SEND` or `IF-RECV` for a given connection, such as an `IF-SEND` following an `IF-SEND` or an `IF-RECV` without a preceding `IF-SEND`, the Responder shall return an `SPDM Storage Protocol Error` of `Invalid or unsupported value in command`.

142 The status for SPDM Storage Messages is explained in Status response hierarchy.

143 5.5.1 SPDM Storage Message `IF-SEND`

144 An `IF-SEND` with the `SPDMOperation` field set to `SPDM Storage Message`, as Table 2 — SPDM Storage Operation Codes defines, shall be used to send an SPDM request from an initiator to a target. The data buffer shall contain request data that is of length `Transfer Length`, including any pad bytes.

145 The data buffer transmitted from the initiator to the target shall be an SPDM request.

146 5.5.2 SPDM Storage Message `IF-RECV`

147 An `IF-RECV` with the `SPDMOperation` field set to `SPDM Storage Message`, as Table 2 — SPDM Storage Operation Codes defines, shall be used to transfer an SPDM response from a target to an initiator. The response data shall be in a data buffer, the size of which is less than or equal to the value in the Allocation Length field.

148 The value in the `ConnectionID` field in the `SPDM Storage Message` `IF-RECV` shall be the same as the value in the `ConnectionID` field in the corresponding `SPDM Storage Message` `IF-SEND`.

149 The data buffer transmitted from the target to the initiator shall be the SPDM response data, including any SPDM `ERROR` response data.

150 5.5.3 Encapsulated request flow

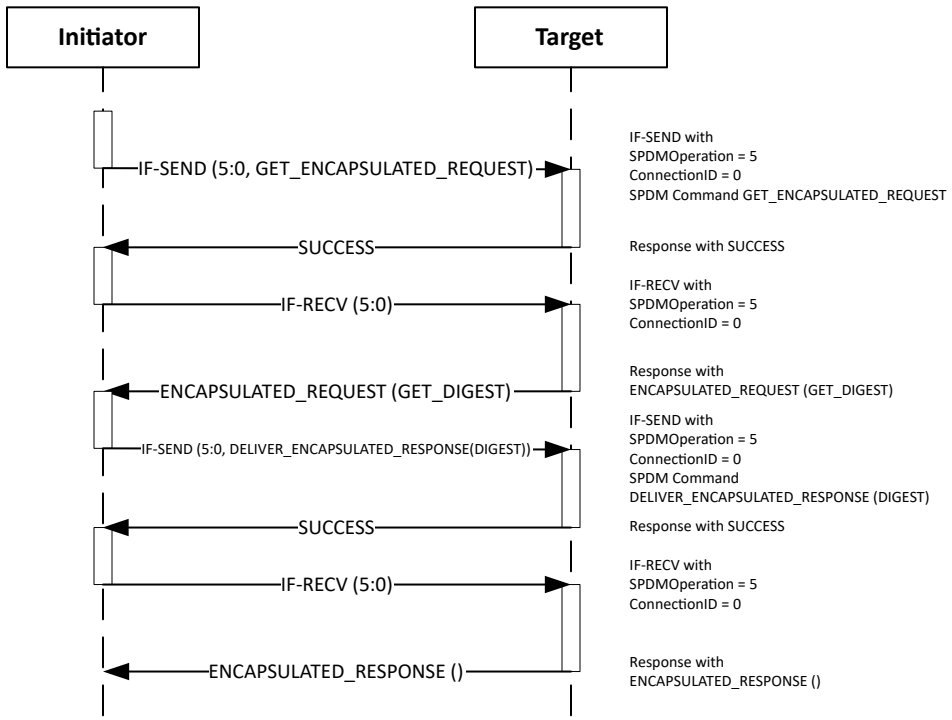
151 Since the SPDM Storage Binding does not support asynchronous requests from a target to an initiator, any message

flows that require a target to send a request to the initiator shall use the SPDM Encapsulated Request Flow. DSP0274 defines the SPDM Encapsulated Request Flow, and the flow uses the `SPDM Storage Message`.

152 Figure 3 — Storage encapsulated message flow shows the SPDM encapsulated message flow for a storage protocol. When an initiator starts an encapsulated flow, the initiator shall use the same `ConnectionID` for all messages in the encapsulated flow.

153 **Figure 3 — Storage encapsulated message flow**

154



155

5.6 SPDM Storage Secured Message

156 The `SPDM Storage Secured Message` uses `IF-SEND` and `IF-RECV` to send and receive Secured Messages, as DSP0277 defines, between an initiator and a target. The `SPDM Storage Secured Message` uses descriptors and buffers to describe the messages and/or data that are transported by the Secured Message. The data buffer for an `SPDM Storage Secured Message` shall follow the format shown in Figure 4 — SPDM Storage Secured Message data buffer and described in this clause. When an initiator sends a Secured Message to a target, the initiator shall use `IF-SEND` with the buffer format described in this clause. When a target responds to an initiator, the target shall fill the buffer supplied by `IF-RECV` with data in the buffer format described in this clause.

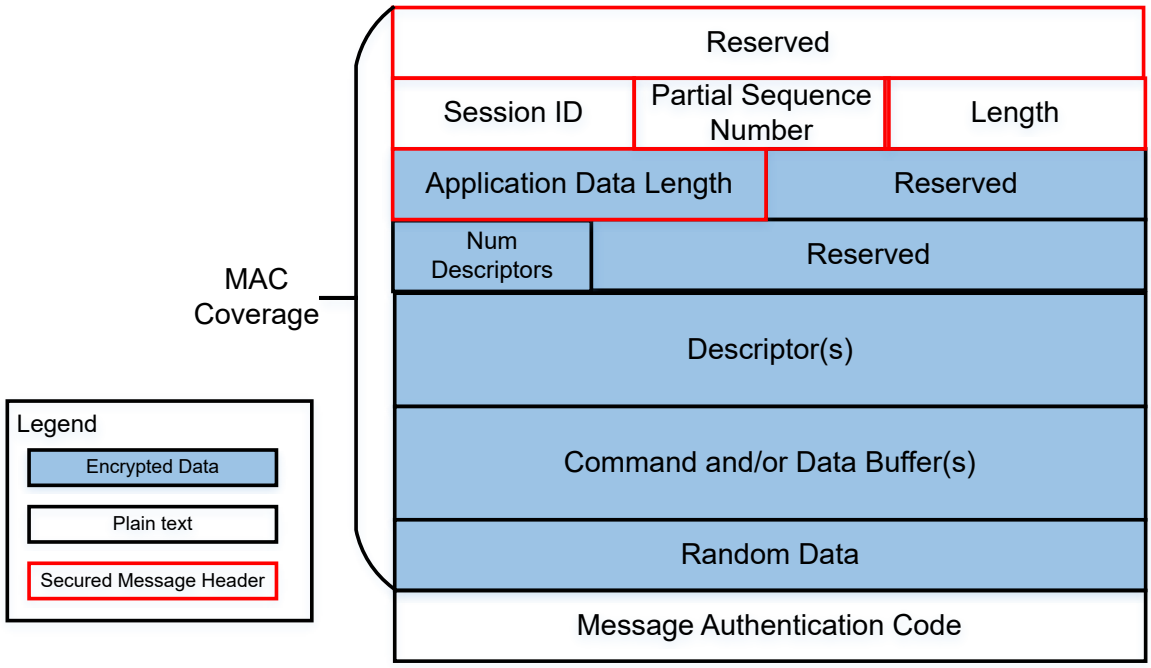
157 The `SPDM Storage Secured Message` shall be restricted to the encapsulation of `IF-SEND` and `IF-RECV` commands. An `SPDM Storage Secured Message` shall not encapsulate another `SPDM Storage Secured Message`. If a target receives an

`SPDM Storage Secured Message` with an encapsulated command that is not permitted, the target shall respond with an `SPDM Storage Protocol Error` of `Invalid Encapsulated Parameter`.

158 If encryption is enabled, the message data is encrypted from `Application Data Length` through `Random Data`, inclusive. The Message Authentication Code calculation covers the fields from the `Reserved` at offset 0 through `Random Data`, inclusive. The fields from `Num Descriptors` through `Command or Data Buffer`, inclusive, shall be treated as the `Application Data` as described by DSP0277.

159 **Figure 4 — SPDM Storage Secured Message data buffer**

160



161 Note: Figure 4 — SPDM Storage Secured Message data buffer is not drawn to scale.

162 Table 7 — SPDM Storage Secured Message data buffer field descriptions describes the fields shown in Figure 4 — SPDM Storage Secured Message data buffer.

163 **Table 7 — SPDM Storage Secured Message data buffer field descriptions**

Byte offset	Field	Size (bytes)	Description
0	Reserved	4	Reserved.
4	`Session ID`	4	Shall be the `Session ID` field as DSP0277 defines.
8	`Partial Sequence Number`	2	Shall be the lower 16 bits of the sequence number as DSP0277 defines.

Byte offset	Field	Size (bytes)	Description
10	`Length`	2	Shall be the `Length` field as DSP0277 defines.
12	`Application Data Length`	2	Shall be the `Application Data Length` field as DSP0277 defines.
14	Reserved	2	Reserved.
16	`Num Descriptors`	1	Shall be the number of `SPDM Storage Secured Message Descriptor` elements in this SPDM Storage Secured Message data buffer.
17	Reserved	3	Reserved.
20	`SPDM Storage Secured Message Descriptor`	16 * `Num Descriptors`	Shall contain `Num Descriptors` instances of the `SPDM Storage Secured Message Descriptor`s.
20 + 16 * `Num Descriptors`	`Secure Message Data`	Variable	Shall be `Num Descriptors` buffers that contain data described by the `SPDM Storage Secured Message Descriptor`s. See Table 8 — SPDM Storage Secured Message descriptor format.
Variable	Command or Data Buffer	Variable	Shall be the command and/or data buffer described by the `SPDM Storage Secured Message Descriptor`s.
Variable	`Random Data`	Variable	Shall be the `Random Data` field as DSP0277 defines.
Variable	`Message Authentication Code` (MAC)	Variable	Shall be the `Message Authentication Code` field as DSP0277 defines.

164 Table 8 — SPDM Storage Secured Message descriptor format describes the `SPDM Storage Secured Message Descriptor` format.

165 **Table 8 — SPDM Storage Secured Message descriptor format**

Byte offset	Field	Size (bytes)	Description
0	Reserved	1	Reserved.
1	`DescType`	1	Shall be the type of the descriptor element, as Table 9 — SPDM Storage Secured Message descriptor types defines. An `SPDM Storage Secured Message` shall contain exactly one descriptor from the `Command` category and may contain up to one descriptor from the `Data Buffer` category. Because SPDM does not use an associated data buffer, a descriptor of type `SPDM` shall not include an associated data buffer.

Byte offset	Field	Size (bytes)	Description
2	`Status`	1	In an SPDM storage secured message request, this field shall be reserved. In an SPDM storage secured message response, this field shall be populated as Secured message encapsulated status defines.
3	Reserved	1	Reserved.
4	`Length`	4	Shall be the length of the corresponding `Secure Message Data` element.
8	`Offset`	4	Shall be the offset of the corresponding `Secure Message Data` element from the start of the `SPDM Storage Secured Message` data buffer. If the `Length` and/or `Offset` for one `Secure Message Data` element overlaps with any other `Secure Message Data` element in this message, the device shall return an `SPDM Storage Protocol Error` of `Invalid Encapsulated Parameter`. See SPDM Storage protocol status.
12	Reserved	4	Reserved.

166 Table 9 — SPDM Storage Secured Message descriptor types describes the details of the `DescType` field in the `SPDM Storage Secured Message Descriptor`.

167 **Table 9 — SPDM Storage Secured Message descriptor types**

Descriptor Type	`DescType` Value	Category	Description
NVME	`0x01`	Command	Shall describe an NVMe command as the NVM Express Base Specification describes.
SCSI	`0x02`	Command	Shall describe a SCSI Command Descriptor Block as the SCSI Architectural Model describes.
ATA	`0x03`	Command	Shall describe an ATA command as the ATA Command Set describes.
SPDM	`0x04`	Command	Shall describe an SPDM request or response message as DSP0274 describes. This `DescType` is used to transmit SPDM commands in a secured session.
DataBuffer	`0x40`	Data Buffer	Shall describe a data transfer buffer. The data transfer buffer shall immediately follow the command with which it is associated.
Reserved	All others	Reserved	All others reserved.

168 ### 5.6.1 Use of descriptors

169 The use of `SPDM Storage Secured Message Descriptor`s in the `SPDM Storage Secured Message` allows a single Secured Message to transmit all elements of a single command to an endpoint. A Requester shall send only one `SPDM Storage Secured Message` per connection at a time and shall wait for a response or timeout before sending the next `SPDM Storage Secured Message` in that connection. The use of descriptors shall not allow endpoints to bypass any message restrictions outlined in SPDM (DSP0274) or Secured Messages (DSP0277).

170 If a device cannot process the provided command or data buffer, the device shall return an appropriate error defined by the protocol specification.

171 ### 5.6.2 Sequence number

172 The sequence number shall be the full width as described in DSP0277. Because only the lower 16 bits of the sequence number is transmitted in the Partial Sequence Number field, the upper 48 bits of the sequence number shall be internally tracked.

173 Because part of the sequence number is transmitted, there may be additional actions that the receiver of the data needs to take. To avoid replay attacks, the receiver of a Secured Message should discard messages with sequence numbers that have already been successfully authenticated and decrypted. See DTLS 1.3 for further guidance.

174 # 6 Storage specific accommodations

175 This clause describes behaviors that are unique to the use of SPDM on storage interfaces and that are generalized across all supported storage interfaces.

176 ## 6.1 Transcript hash calculation

177 Transcript hashes shall be calculated as DSP0274 describes. Transcript hashes shall be calculated over the contents of `SPDM Storage Message` data buffers but shall not include the fields in `IF-SEND` and `IF-RECV`. Transcript hashes shall not include any pad data from the data buffer.

178 ### 6.1.1 Padded transactions

179 Certain storage protocols require or support specifying the length of `IF-SEND` and `IF-RECV` data buffers in increments of either 4 bytes or 512 bytes. If a command specifies a data buffer size that is larger than the message that is being transmitted, the buffer requires a pad at the end of the buffer. In such a case, the SPDM message shall start from Byte[0] of the data buffer. The pad bytes shall fill from after the end of the SPDM message until the end of the data buffer. For `IF-SEND`, the Requester shall use `0x00` for the pad value and the Responder shall ignore the pad value. For `IF-RECV`, the Responder shall use `0x00` for the pad value and the Requester shall ignore the pad value. See Transcript hash calculation for implications of pad values on transcript hash calculations.

180 Figure 5 — SCSI SECURITY PROTOCOL OUT CDB shows a SCSI SECURITY PROTOCOL OUT (generically an IF-SEND) Command Descriptor Block (CDB) to illustrate the concept of a padded transaction. This figure shows an `SPDM Storage Message` and the INC_512 flag is set, meaning that the data buffer must be allocated in 512-byte increments. The TRANSFER LENGTH field is set to 1 to indicate that the data buffer is 512 bytes. None of the bytes in the SCSI SECURITY PROTOCOL OUT CDB are included in the transcript hash calculation.

181 **Figure 5 — SCSI SECURITY PROTOCOL OUT CDB**

Bit / Byte	7	6	5	4	3	2	1	0
0	0xB5 (SECURITY PROTOCOL OUT)							
1	0xE8 (SPDM Security Protocol)							
2 3	0x0020 (0x05 - SPDM Storage Message, 0x0 - ConnectionID)							
4	0x80 (INC_512 = 1b)							
5	Reserved							
6 ... 9	0x00000001 (TRANSFER LENGTH = 512 bytes)							
10	Reserved							
11	CONTROL							

182
183 Figure 6 — Padded SPDM GET_DIGESTS shows a data buffer that contains an SPDM GET_DIGESTS request. In this figure, the `SPDMVersion` field in the request is set to 0x13 to indicate that DSP0274 version 1.3 is in use, but any supported DSP0274 version can be used. In Figure 6 — Padded SPDM GET_DIGESTS, bytes 0 through 3 inclusive (shown in the figure with red outlines) are included in the transcript hash, whereas Bytes 4 through 511 inclusive (shown in the figure with gray background) are not included in the transcript hash because they are pad bytes.

184 **Figure 6 — Padded SPDM GET_DIGESTS**

185

Byte Offset	3	2	1	0
0	0x00 (Reserved)	0x00 (Reserved)	0x81 (GET_DIGESTS)	0x13 (SPDMVersion)
4	0x00000000			
8	0x00000000			
...	0x00000000			
508	0x00000000			

Legend:

Included in Transcript Hash

Pad Bytes - not in Transcript Hash

6.2 GET_VERSION handling

DSP0274 specifies that the `GET_VERSION` request restarts the SPDM protocol. A `GET_VERSION` request can break the sequencing rules described in SPDM Storage Message, so a `GET_VERSION` request can be sent while an `IF-SEND` is outstanding.

In addition to the `GET_VERSION` behaviors defined in DSP0274, when a Responder receives a `GET_VERSION` request it shall clear any data associated with `SPDM Storage Pending Info`.

6.3 Device reset handling

Each storage interface or its underlying transport defines one or more reset operations. Storage interface specific behavior specifies SPDM behaviors related to resets. Storage interface and underlying transport resets that are not specified by this specification shall not have an effect on the SPDM protocol behavior. For interface-specific resets that result in a `Device reset`, the device shall process the reset according to the Reset definition in DSP0274.

At power-on reset, a device shall clear all SPDM state machines, connections, and secure sessions, with the exception that, if supported, data may be cached by the devices, such as `VCA` data. See DSP0274 for more details.

6.4 Status response hierarchy

When using the commands defined by this specification, error responses can occur in multiple domains, or at multiple layers of the interface between the Requester and Responder. The device shall report errors in the following domain order, and the device is not required to report status from a lower-order domain when a higher-order domain has a non-successful completion status. The following list gives the order in which error domains are processed, with the lowest-numbered domain being given highest order.

1. Interface status. Interface status shall encompass all statuses that indicate whether the request or response was correctly received by the other device. Interface errors can include, but are not limited to, errors at the physical, link, or transport layer. These errors shall be reported using protocol or transport-defined error messages.
2. SPDM Storage protocol status. The domain of SPDM Storage protocol status shall define status reporting, as the SPDM Storage protocol status clause defines, for errors that are detected in the format or contents of an `IF-SEND` or `IF-RECV` that contains an SPDM Storage protocol command, as Table 2 — SPDM Storage Operation Codes defines.
3. SPDM protocol status. SPDM protocol status shall report status as DSP0274 and DSP0277 define. Any hierarchy or prioritization of error status reporting between these specifications shall be as these specifications define. For example, an SPDM `ERROR` response would be associated with a successful status for the storage transport, unless a protocol condition occurs in conjunction with the SPDM `ERROR` response.
4. Secured message encapsulated status. Status that is reported in response to a command sequence that occurs in an SPDM Storage Secured Message session shall be reported in a session using a

Secured message encapsulated error. See Secured message encapsulated status.

6.4.1 SPDM Storage protocol status

This specification refers to errors that are in the SPDM Storage protocol status category as `SPDM Storage Protocol Error`s. Devices shall report `SPDM Storage Protocol Error`s using protocol-specific error codes, as specified in Storage interface specific behavior. A Responder shall send status using the same protocol as the protocol used for the associated request.

Table 10 — SPDM Storage Protocol Errors describes the `SPDM Storage Protocol Error` conditions. Storage interface specific behavior maps specific `SPDM Storage Protocol Error` conditions to underlying interface status codes.

Table 10 — SPDM Storage Protocol Errors

SPDM Storage Protocol Error	Comments
Success	Indicates successful transmission of the message.
Invalid or unsupported value in command	At least one field in the `IF-SEND` of `IF-RECV` contains an unknown or unsupported value.
Invalid Transfer Length in `IF-SEND`	Only valid with `IF-SEND`. The data buffer does not contain a complete message.
Invalid Encapsulated Parameter	At least one field in the SPDM Storage Secured Message data buffer contains an unknown or unsupported value. This error indicates that the format of the data buffer could not be interpreted. If the SPDM Secured Message is properly interpreted, but causes an error, the error shall reported as Secured message encapsulated status defines.

6.4.2 Secured message encapsulated status

The status of an encapsulated command in a secured message is returned using an `SPDM Storage Secured Message Descriptor`.

If the encapsulated command is an SPDM Request, then the status shall be returned in an `SPDM Message`. This response takes the form of either an SPDM-formatted response to the request or an `ERROR` response, as DSP0274 and DSP0277 define.

If the encapsulated command is an NVMe, SCSI, or ATA command, the response status shall be returned in the `Status` field of a `Data Buffer` type `SPDM Storage Secured Message Descriptor` as Table 8 — SPDM Storage Secured Message descriptor format defines. In this case, the `Status` field shall contain a `StatusCode` as Table 11 — Encapsulated Response Status Codes defines. If the `StatusCode` is any value other than `Success`, the `Data Buffer`

pointed to by the `SPDM Storage Secured Message Descriptor` shall only contain `StorageErrorData`, if any is defined. If `StorageErrorData` is not defined for the given `StatusCode`, the `Length` and `Offset` fields of the `Data Buffer` type `SPDM Storage Secured Message Descriptor` shall be set to `0`.

206 **Table 11 — Encapsulated Response Status Codes**

StatusCode Value	StatusCode	StorageErrorData	Description
`0x00`	Success	None	The command completed without an error.
`0x01`	General Error	None	The device encountered an error while processing the request. Protocol-specific mechanisms may contain additional information about the error.
`0x02`	Invalid Command	None	The command code is not recognized or not supported in an SPDM storage secured message.
`0x03`	Invalid Field	None	The command contains one or more fields that are not valid.
`0xFF`	Vendor Defined	See Vendor-defined secured message encapsulated status	The format of the Vendor Defined StatusCode shall be as Vendor-defined secured message encapsulated status defines.

207 **6.4.2.1 Vendor-defined secured message encapsulated status**

208 The format for the `Vendor Defined` `Secured Message Encapsulated Status` shall be as Table 12 — Vendor-defined secured message encapsulated status defines. This format follows the `SVH` format as defined in DSP0274.

209 **Table 12 — Vendor-defined secured message encapsulated status**

Offset	Field	Length (bytes)	Description
0	ID	1	Shall be one of the values in the `ID` column of "Registry or standards body ID" table as defined in DSP0274.

Offset	Field	Length (bytes)	Description
1	`VendorLen`	1	Length in bytes of the `VendorID` field. If the definition of the data in `ErrorData` belongs to a standards body, this field shall be 0. Otherwise, the definition of the data in `ErrorData` belongs to the identified vendor and therefore, this field shall be the length indicated in the `Vendor ID` column of "Registry and standards body ID" table for the respective `ID` defined in DSP0274.
2	`VendorID`	`VendorLen`	If `VendorLen` is greater than zero, this field shall be the ID of the vendor corresponding to the `ID` field. Otherwise, this field shall be absent.
2 + `VendorLen`	`ErrorDataLen`	2	Shall be the length, in bytes, of the `ErrorData` field.
2 + `VendorLen` + `ErrorDataLen`	`VenErrorData`	`ErrorDataLen`	Shall contain the vendor or standards body defined error data.

6.5 Multi-path handling

211 A Requester should use protocol-specific commands to discover the relationship between different ports on the same device. When a Responder supports multiple paths (i.e., multiple routes over the transport to the same endpoint), SPDM-defined information in the payload that is presented by the Responder shall be the same for all paths unless the change in the information reflects the path differences. The connection between a given Requester and a specific

Responder port shall be considered a unique SPDM connection. SPDM sessions shall be restricted to the port on which they were negotiated, so therefore separate device ports require separate sessions.

212 Response data, such as certificate chains and measurements, should be the same regardless of which port is used to retrieve the data. In some cases, a response may include port-specific information, in which case the information is required to be specific to the port through which it is retrieved.

7 Storage interface specific behavior

This clause specifies behaviors that differ between storage interfaces and specifies the specific behavior for each supported interface.

7.1 NVMe specific behavior

NVMe specific behavior includes use of the NVMe interface over PCIe®, RDMA, and TCP transports.

7.1.1 NVMe byte ordering

NVMe specifies that NVMe defined fields are little endian. Storage protocol level fields in this specification shall follow the endianness rules for the protocol in use. SPDM-specified fields shall follow the endianness rules for the relevant SPDM specification.

7.1.2 NVMe command format

7.1.2.1 `Security Send` format

Table 13 — `Security Send` field definitions describes the use of fields in the NVMe `Security Send` command. The use of fields not defined in the following table shall conform to the behavior defined in the NVMe specification.

Table 13 — `Security Send` field definitions

Field	Usage
Security Protocol	This field shall be set to `0xE8` for SPDM commands.
SP Specific 1	Reserved.
SP Specific 0	Bit [1:0]. Shall contain the `ConnectionID` for the command. Bit [7:2]. Shall contain the `SPDMOperation` field, as Table 2 — SPDM Storage Operation Codes defines.
Transfer Length	This field shall be the length of the data buffer to be transferred from the Requester to the Responder, inclusive of any required padding.

7.1.2.2 `Security Receive` format

Table 14 — `Security Receive` field definitions describes the use of fields in the NVMe `Security Receive` command. The use of fields not defined in the following table shall conform to the behavior defined in the NVMe specification.

225 **Table 14 — `Security Receive` field definitions**

Field	Usage
Security Protocol	This field shall be set to `0xE8` for SPDM commands.
SP Specific 1	Reserved.
SP Specific 0	Bit [1:0]. Shall contain the `ConnectionID` for the command. Bit [7:2]. Shall contain the `SPDMOperation` field, as Table 2 — SPDM Storage Operation Codes defines.
Allocation Length	This field shall be the length of the data buffer to receive data from the Responder to the Requester, inclusive of any required padding.

226 #### 7.1.2.3 Namespace addressing

227 When the `SECURITY PROTOCOL` field is set to `0xE8` for an NVMe device, the use of the Namespace Identifier (NSID) field shall be reserved.

228 #### 7.1.2.4 NVMe device reset handling

229 When using the NVMe protocol, the mapping of resets to SPDM behavior shall follow the behavior that Table 15 — NVMe reset to SPDM mapping defines.

230 **Table 15 — NVMe reset to SPDM mapping**

NVMe Reset Event	SPDM Behavior
NVMe controller reset	Device reset.
NVMe subsystem reset	Device reset.

231 #### 7.1.2.5 NVMe protocol status

232 Table 16 — NVMe reporting of SPDM Storage Protocol Errors defines the reporting of `SPDM Storage Protocol Error`s using the NVMe protocol.

233 **Table 16 — NVMe reporting of SPDM Storage Protocol Errors**

SPDM Storage Protocol Error	NVMe Status Code Type	NVMe Status Code	NVMe Do Not Retry Bit	Comments
Success	Generic Command Status	Successful Completion	0	Normal command completion.
Invalid or unsupported value in command	Generic Command Status	Invalid Field in Command	1	No data shall be transferred.

SPDM Storage Protocol Error	NVMe Status Code Type	NVMe Status Code	NVMe Do Not Retry Bit	Comments
Invalid Transfer Length in `IF-SEND`	Generic Command Status	Invalid Field in Command	1	No data shall be transferred.
Invalid Encapsulated Parameter	Generic Command Status	Invalid Field in Command	1	No data shall be transferred.

234 ### 7.1.3 NVMe multi-path handling

235 NVMe devices can support multi-path configurations, as Multi-path handling describes.

236 ## 7.2 SCSI specific behavior

237 SCSI specific behavior includes use of the SCSI interface over SAS, Fibre Channel, ATAPI, UAS, USB, and UFS transports.

238 ### 7.2.1 SCSI byte ordering

239 SAM-6 specifies that SCSI-defined fields are big endian. Storage protocol level fields in this specification shall follow the endianness rules for the protocol in use. SPDM-specified fields shall follow the endianness rules for the relevant SPDM specification.

240 ### 7.2.2 SCSI command format

241 #### 7.2.2.1 `SECURITY PROTOCOL OUT` format

242 Table 17 — `SECURITY PROTOCOL OUT` field definitions describes the use of fields in the SCSI `SECURITY PROTOCOL OUT` command. The use of fields not defined in the following table shall conform to the behavior defined in the SCSI specification.

243 **Table 17 — `SECURITY PROTOCOL OUT` field definitions**

Field	Usage
SECURITY PROTOCOL	This field shall be set to `0xE8` for SPDM commands.
SECURITY PROTOCOL SPECIFIC	Bit [1:0]. Shall contain the `ConnectionID` for the command. Bit [7:2]. Shall contain the `SPDMOperation` field, as Table 2 — SPDM Storage Operation Codes defines. All other values reserved.
INC_512	This bit is managed as SPC-6 describes.
TRANSFER LENGTH	This field shall be the length of the data buffer to be transferred from the Requester to the Responder, inclusive of any required padding.

244 #### 7.2.2.2 `SECURITY PROTOCOL IN` format

245 Table 18 — `SECURITY PROTOCOL IN` field definitions describes the use of fields in the SCSI `SECURITY PROTOCOL IN` command. The use of fields not defined in the following table shall conform to the behavior defined in the SCSI specification.

246 **Table 18 — `SECURITY PROTOCOL IN` field definitions**

Field	Usage
SECURITY PROTOCOL	This field shall be set to `0xE8` for SPDM commands.
SECURITY PROTOCOL SPECIFIC	Bit [1:0]. Shall contain the `ConnectionID` for the command. Bit [7:2]. Shall contain the `SPDMOperation` field, as Table 2 — SPDM Storage Operation Codes defines. All other values reserved.
INC_512	This bit is managed as SPC-6 describes.
ALLOCATION LENGTH	This field shall be the length of the data buffer to receive data from the Responder to the Requester, inclusive of any required padding.

247 #### 7.2.2.3 Logical unit addressing

248 When the `SECURITY PROTOCOL` field is set to `0xE8` for a SCSI device, the use of a LOGICAL UNIT NUMBER other than LUN 0 or the SECURITY PROTOCOL well-known logical unit shall be reserved.

249 #### 7.2.2.4 SCSI device reset handling

250 When using the SCSI protocol on a SAS transport, the mapping of resets to SPDM behavior shall follow the behavior that Table 19 — SAS transport reset to SPDM mapping defines.

251 **Table 19 — SAS transport reset to SPDM mapping**

SAS Reset Event	SPDM Behavior
LOGICAL UNIT RESET task management function	Device reset.
Link reset sequence with hard reset	Device reset.

252 #### 7.2.2.5 SCSI protocol status

253 Table 20 — SCSI reporting of SPDM Storage Protocol Errors defines the reporting of `SPDM Storage Protocol Error`s using the SCSI protocol.

254 **Table 20 — SCSI reporting of SPDM Storage Protocol Errors**

SPDM Storage Protocol Error	SCSI Status	SCSI Sense Key	SCSI ASC/ASCQ	Comments
Success	GOOD	NO SENSE	NO ADDITIONAL SENSE INFORMATION	Normal command completion.
Invalid or unsupported value in command	CHECK CONDITION	ILLEGAL REQUEST	INVALID FIELD IN CDB	No data shall be transferred.
Invalid Transfer Length in `IF-SEND`	CHECK CONDITION	ILLEGAL REQUEST	INVALID FIELD IN CDB	No data shall be transferred.
Invalid Encapsulated Parameter	CHECK CONDITION	ILLEGAL REQUEST	INVALID FIELD IN PARAMETER LIST	No data shall be transferred.

255 ### 7.2.3 SCSI multi-path handling

256 SCSI devices can support multi-path configurations, as Multi-path handling describes.

257 ## 7.3 ATA specific behavior

258 ATA specific behavior includes use of the ATA interface over SATA, PATA, CFast® , and CompactFlash® transports.

259 ### 7.3.1 ATA byte ordering

260 ACS-5 specifies that ATA defined fields are big endian. Storage protocol level fields in this specification shall follow the endianness rules for the protocol in use. SPDM-specified fields shall follow the endianness rules for the relevant SPDM specification.

261 ### 7.3.2 ATA command format

262 #### 7.3.2.1 `TRUSTED SEND` and `TRUSTED SEND DMA` format

263 Table 21 — `TRUSTED SEND` and `TRUSTED SEND DMA` field definitions describes the use of fields in the ATA `TRUSTED SEND` and `TRUSTED SEND DMA` command. The use of fields not defined in the following table shall conform to the behavior defined in the ATA specification.

264 **Table 21 — `TRUSTED SEND` and `TRUSTED SEND DMA` field definitions**

Field	Usage
SECURITY PROTOCOL	This field shall be set to `0xE8` for SPDM commands.

Field	Usage
SP SPECIFIC	Bit [1:0]. Shall contain the `ConnectionID` for the command. Bit [7:2]. Shall contain the `SPDMOperation` field, as Table 2 — SPDM Storage Operation Codes defines. All other values reserved.
TRANSFER LENGTH	This field shall be the length of the data buffer to be transferred from the Requester to the Responder, inclusive of any required padding.

265 #### 7.3.2.2 `TRUSTED RECEIVE` and `TRUSTED RECEIVE DMA` format

266 Table 22 — `TRUSTED RECEIVE` and `TRUSTED RECEIVE DMA` field definitions describes the use of fields in the ATA `TRUSTED RECEIVE` and `TRUSTED RECEIVE DMA` command. The use of fields not defined in the following table shall conform to the behavior defined in the ATA specification.

267 **Table 22 — `TRUSTED RECEIVE` and `TRUSTED RECEIVE DMA` field definitions**

Field	Usage
SECURITY PROTOCOL	This field shall be set to `0xE8` for SPDM commands.
SP SPECIFIC	Bit [1:0]. Shall contain the `ConnectionID` for the command. Bit [7:2]. Shall contain the `SPDMOperation` field, as Table 2 — SPDM Storage Operation Codes defines. All other values reserved.
ALLOCATION LENGTH	This field shall be the length of the data buffer to receive data from the Responder to the Requester, inclusive of any required padding.

268 #### 7.3.2.3 ATA device reset handling

269 When using the ATA protocol, the mapping of resets to SPDM behavior shall follow the behavior that Table 23 — ATA protocol reset to SPDM mapping defines.

270 **Table 23 — ATA protocol reset to SPDM mapping**

ATA Reset Event	SPDM Behavior
COMRESET with Software Settings Preservation disabled	Device reset.

271 #### 7.3.2.4 ATA protocol status

272 Table 24 — ATA reporting of SPDM Storage Protocol Errors without SDR defines the reporting of `SPDM Storage Protocol Error`s using the ATA Command Set when Sense Data Reporting (SDR) is not available (due to SDR not being available, SDR being in a disabled state, or the `SENSE DATA AVAILABLE` bit is set to `0`).

273 **Table 24 — ATA reporting of SPDM Storage Protocol Errors without SDR**

SPDM Storage Protocol Error	ATA Status Field	ATA Error Field	Comments
Success	`0x50`	`0x00`	Normal command completion.
Invalid or unsupported value in command	`0x51`	`0x04`	No data shall be transferred.
Invalid Transfer Length in `IF-SEND`	`0x51`	`0x04`	No data shall be transferred.
Invalid Encapsulated Parameter	`0x51`	`0x04`	No data shall be transferred.

274 If the device supports SDR and if SDR is enabled and if the `SENSE DATA AVAILABLE` bit is set to `1`, then status shall be reported as Table 20 — SCSI reporting of SPDM Storage Protocol Errors defines, with the addition that Bit[1] of the `ATA Status Field` shall be set to `1`.

275 ### 7.3.3 ATA multi-path handling

276 ATA devices do not support multi-path configurations.

8 ANNEX A (informative) Change log

8.1 Version 1.0.0 (2025-05-15)

- Initial release

9 ANNEX B (informative) Bibliography

281 DMTF DSP4014, *DMTF Process for Working Bodies*, https://www.dmtf.org/sites/default/files/standards/documents/DSP4014.pdf