1 # Secured Messages using SPDM Specification

2 # Version: 2.0.0

3 **Document Identifier: DSP0277**

4 **Date: 2025-10-30**

5 **Version History: https://www.dmtf.org/dsp/DSP0277**

6 **Supersedes: None**

7 **Document Class: Normative**

8 **Document Status: Published**

9 **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patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a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 This document's normative language is English. Translation into other languages is permitted.

CONTENTS

14 1 Foreword

15 The Security Protocols and Data Models (SPDM) Working Group of DMTF prepared the *Secured Messages using SPDM Specification* (DSP0277). DMTF is a not-for-profit association of industry members that promotes enterprise and systems management and interoperability. For information about DMTF, see dmtf.org.

16 1.1 Acknowledgments

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

18 **Contributors:**

- Steven Bellock — NVIDIA Corporation
- Patrick Caporale — Lenovo
- Nigel Edwards — Hewlett Packard Enterprise
- Daniil Egranov — Arm Limited
- Philip Hawkes — Qualcomm Inc.
- Brett Henning — Broadcom Inc.
- Jeff Hilland — Hewlett Packard Enterprise
- Yuval Itkin — NVIDIA Corporation
- Theo Koulouris — Hewlett Packard Enterprise
- Eliel Louzoun — Intel Corporation
- Donald Matthews — Advanced Micro Devices, Inc.
- Edward Newman — Hewlett Packard Enterprise
- Jim Panian — Qualcomm Inc.
- Scott Phuong — Cisco Systems Inc., Axiado Corporation, Microsoft Corporation
- Viswanath Ponnuru — Dell Technologies
- Xiaoyu Ruan — Intel Corporation
- Nitin Sarangdhar — DMTF
- Bob Stevens — Dell Technologies
- Jiewen Yao — Intel Corporation
- Wilson Young — Solidigm

19 # 2 Introduction

20 The *Secured Messages using SPDM Specification* defines the methodology that various PMCI transports can use to communicate various application data securely by utilizing SPDM. Specifically, this specification defines the transport requirements for SPDM records, which form the basis of encryption and message authentication.

21 Furthermore, this specification contains guidance and certain decisions that are deferred to the transport binding specification, which binds Secured Messages to a specific transport. Thus, the transport binding specification is expected to finalize those decisions or guidance by way of normalization or recommendation. The present specification was written with PMCI transports in mind, but nothing precludes specifying bindings to other transports.

22 ## 2.1 Motivation for version 2.0

23 The version 2.0 specification contains major changes in the header format that are not compatible with the version 1 releases of this specification. These changes were introduced to support authorization and to facilitate future expansion of the header fields.

24 For details of changes in released versions, see Annex B (informative) change log.

25 ## 2.2 Compatibility

26 This specification is decoupled from DSP0274 to avoid unnecessary updates here whenever SPDM changes. If a tighter coupling to DSP0274 is desired, the transport binding specification should describe it.

27 Furthermore, this specification follows these rules to minimize changes between minor versions to ensure compatibility:

- Computations and other operations between different minor versions of the Secured Messages using SPDM specification should remain the same, unless security issues of lower minor versions are fixed in higher minor versions and the fixes require changes in computations or other operations.
- In a newer minor version of the Secured Messages using SPDM specification, a given message can be longer, bit fields and enumerations can contain new values, and reserved fields can gain functionality. Existing numeric and bit fields retain their existing definitions.

28 ## 2.3 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.

29 ## 2.4 Notations

30 This specification uses the following notations:

Notation	Description
`Concatenate()`	The concatenation function `Concatenate(a, b, ..., z)`, where the first entry occupies the least-significant bits and the last entry occupies the most-significant bits.
`M:N`	In field descriptions, this notation typically represents a range of byte offsets starting from byte `M` and continuing to and including byte `N` (`M ≤ N`). The lowest offset is on the left. The highest offset is on the right.
`[4]`	Square brackets around a number typically indicate a bit offset. Bit offsets are zero-based values. That is, the least-significant bit (`[LSb]`) offset = 0.
`[M:N]`	A range of bit offsets where M is greater than or equal to N. The most significant bit is on the left, and the least-significant bit is on the right.
`1b`	A lowercase `b` after a number consisting of `0`s and `1`s indicates that the number is in binary format.
`0x12A`	Hexadecimal, as indicated by the leading `0x`.
`N+`	Variable-length byte range that starts at byte offset N.

`[${message_name}]`.`${field_name}`
 	
or
 	
`[${message_name}]`.`${field_name}`/`${field_name0}`/…/`${field_nameN}`
	Used to indicate a field in a message.
	• `${message_name}` is the name of the request or response message.
	• `${field_name}` is the name of the field in the request or response message. An asterisk (`*`) instead of a field name means all fields in that message except for any conditional fields that are empty.
	• One or more optional forward slash characters (`/`) can follow to indicate hierarchy of field names, similar to a directory path in many operating systems.

Notation	Description
`${field_name}` / `${field_name0}` /…/ `${field_nameN}`	Used to indicate a field in a message relative to `${field_name}`. • `${field_name}` is the name of the field in the request or response message. An asterisk (`*`) instead of a field name means all fields in that message except for any conditional fields that are empty. • One or more optional forward slash characters (`/`) can follow to indicate hierarchy of field names, similar to a directory path in many operating systems.

31 # 3 Scope

32 This document defines a generic record format used to encrypt and authenticate any application data within SPDM's secure session. Also, relating to encryption, message authentication, and secure sessions, this specification further defines those areas in SPDM that the specification states are the responsibilities of the transport layer.

33 This specification requires SPDM version 1.2 or later.

34

4 Normative references

35 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.

- DMTF DSP0274, *Security Protocol and Data Model (SPDM) Specification* 1.2 or later, https://www.dmtf.org/dsp/dsp0274
- DMTF DSP0289, *Authorization Specification* 1.0 or later, https://www.dmtf.org/dsp/dsp0289
- IETF RFC 5116, *An Interface and Algorithms for Authenticated Encryption* (January 2008), https://datatracker.ietf.org/doc/html/rfc5116
- IETF RFC 5234, *Augmented BNF for Syntax Specifications: ABNF* (January 2008), https://datatracker.ietf.org/doc/html/rfc5234
- IETF RFC 8439, *ChaCha20 and Poly1305 for IETF Protocols* (June 2018), https://datatracker.ietf.org/doc/html/rfc8439
- IETF RFC 8998, *ShangMi (SM) Cipher Suites for TLS 1.3* (March 2021), https://datatracker.ietf.org/doc/html/rfc8998
- IETF RFC 9147, *The Datagram Transport Layer Security (DTLS) Protocol* 1.3 (April 2022), https://datatracker.ietf.org/doc/rfc9147
- *ISO/IEC Directives, Part 2, Principles and rules for the structure and drafting of ISO and IEC documents* 9th edition (2021), https://www.iso.org/sites/directives/current/part2/index.xhtml

36 # 5 Terms and definitions

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

38 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.

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

40 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.

41 The terms that DSP0274 defines also apply to this document with the following exception:

- Whereas DSP0274 defines a secure session as consisting of either encryption, message authentication, or both, this document defines a secure session as consisting of both encryption and message authentication or message authentication without encryption.

42
6 Symbols and abbreviated terms

43 The symbols and abbreviations defined in DSP0274 apply to this document.

44 The following additional abbreviations are used in this document.

Abbreviation	Term
LTD	Layered Transport Data

45
7 Secured Messages

46 Starting with SPDM 1.1, SPDM describes at a very high and abstract level a construct, called a record, to encrypt and authenticate data within a session. SPDM places the responsibility for the details and definition of the record on the transport layer. The manifestation of this record in this specification is called a Secured Message.

47 A Secured Message shall be used only within a secure session. Specifically, a Secured Message can be used in any phase of a secure session, such as the session handshake phase and the application phase.

48 To support a Secured Message, an SPDM endpoint shall support message authentication. Additionally, an SPDM endpoint shall support one or more AEAD algorithms as defined in DSP0274. Finally, an SPDM Responder shall select an AEAD algorithm according to DSP0274.

49 Furthermore, a Secured Message can carry data that requires authorization as DSP0289 defines. If an SPDM endpoint supports DSP0289, the SPDM endpoint shall support the Authorization record as DSP0289 defines. Secured Messages within an SPDM session can either directly carry the application data or carry an authorization record that encapsulates the application data.

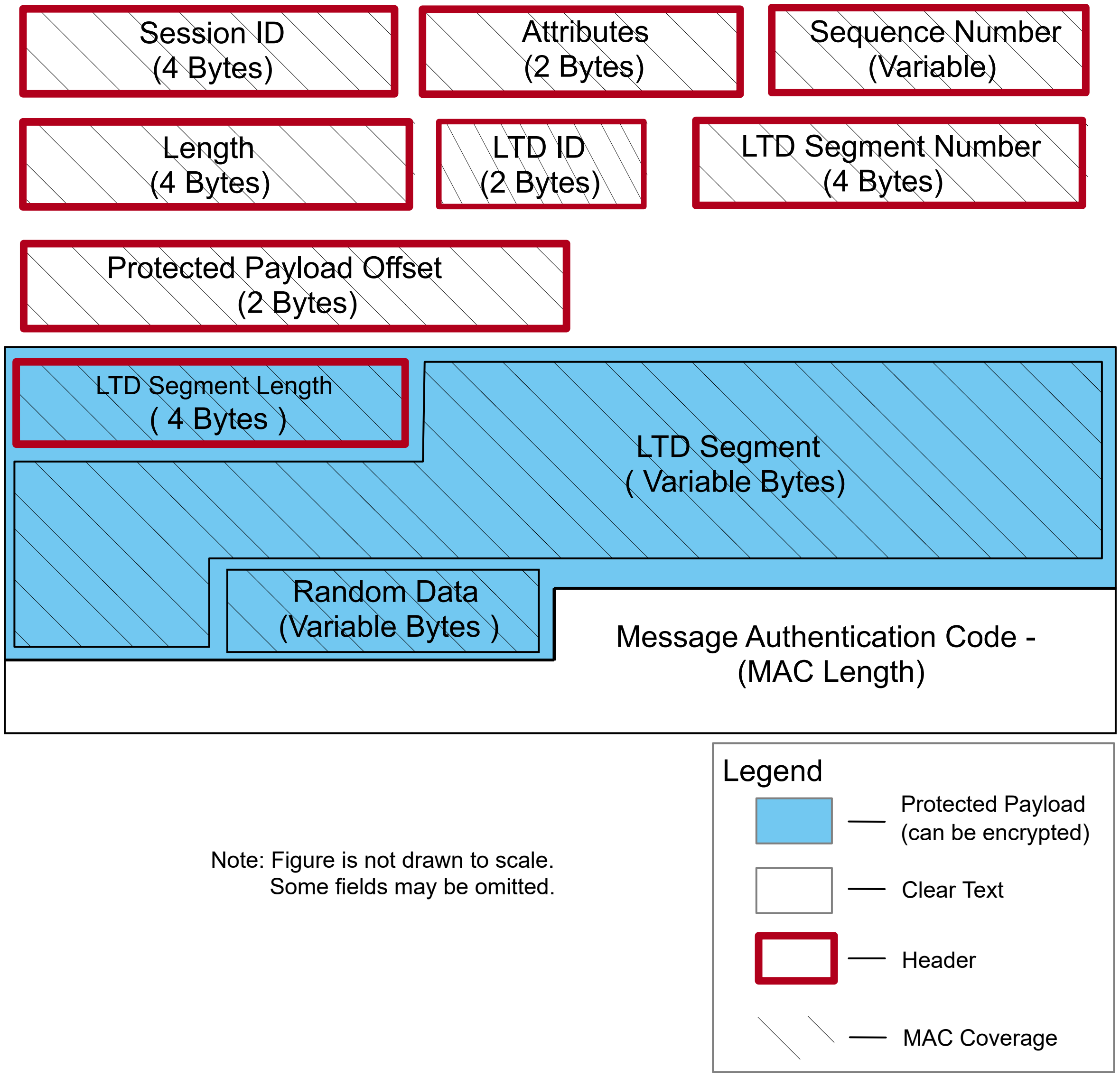
50 When Secured Messages are transmitted between entities that are not SPDM endpoints, the clauses in this specification apply unless otherwise specified by the transport binding.

51
7.1 Secured Message format

52 Figure 1 — Secured Message format illustrates the format used to encrypt, authenticate, or authorize application data. This specification uses a single format for all Layered Transport Data (LTD), such as application data and SPDM messages that are required to be sent within a session such as `KEY_UPDATE` messages. Layered Transport Data is the payload that a Secured Message protects.

53 The Secured Message format figure below illustrates the format of a Secured Message at a high level.

54



55 **Figure 1 — Secured Message format**

56 Table 1 — Secured Message field definitions defines the exact format of a Secured Message. This format is specific to AEAD algorithms that append the MAC to the end of the cipher text. The *Recommendation* column applies to the transport and provides a recommendation of its inclusion in the transport binding specification. A value of *Yes* in this column indicates that the respective field should be included in the transport binding specification. A value of *Only if Required* indicates the respective field should not be included unless necessary and the *Description* column will have further guidance.

57 **Table 1 — Secured Message field definitions**

Offset	Field	Length (bytes)	Recommendation	Description
0	Session ID	4	Yes	The Responder and Requester can use this field to bind all session information such as secrets and keys. This field shall be the same value as the `SessionID` variable in the Session-Secrets-Exchange response. To ensure forward and backward compatibility, this field shall never change and shall always be first.
4	Attributes	2	Yes	This field shall indicate the attributes of the Secured Message. The format of this field shall be as Table 2 — Attributes Bit Definitions defines.
6	Sequence Number	S	Only if Required	This field shall indicate all or part of the Sequence Number of the Secured Messages as described in the Per-message nonce derivation clause. See the Transmission reliability clause for further details. The transport binding specification shall specify the length `S` and further details, if any, for this field.

Offset	Field	Length (bytes)	Recommendation	Description
6 + `S`	Length	4	Yes	This field shall indicate the data length as Additional Secured Message Format Details defines.
10 + `S`	LTD ID	2	Yes	The value of this field shall identify the Layered Transport Data (LTD) as Layered Transport Data Transfer Mechanism defines.
12 + `S`	LTD Segment Number	4	Yes	The value of this field shall indicate the segment number as Layered Transport Data Transfer Mechanism defines.

Offset	Field	Length (bytes)	Recommendation	Description
16 + `S`	Protected Payload Offset	2	Yes	This field points to the starting offset of the encrypted and/or authenticated data relative to this field for the corresponding Secured Message. If `LTD Segment Length` field is present, this field shall indicate the starting offset of the `LTD Segment Length` field relative to the field. Otherwise, this field shall indicate the starting offset of the `LTD Segment` relative to this field. For this version of the specification, the sender of a Secured Message shall populate a value of 0. A value of 0 means the starting offset of the payload portion of the Secured Message is at the absolute offset 18 + `S`. The receiver shall use this value to find the payload portion of this message. This ensures the receiver maintains compatibility in future versions of this specification. When checking the value of this field, the receiver should not check for the expected value in order to maintain

Offset	Field	Length (bytes)	Recommendation	Description
				future compatibility. However, the receiver should perform other checks such as out of boundary checks.
18 + `S`	LTD Segment Length	L	Yes	This field shall be the length of `LTD Segment`. The value of this field shall be greater than zero if present. If present, the size of this field shall be 4; otherwise, this field shall be zero.
22 + `S` + `L`	LTD Segment	P	Yes	This field shall contain Layered Transport Data as indicated by the `Attributes` / `LTDtype` field. See Layered Transport Data Types for further details.
22 + `S` + `L` + `P`	Random Data	R	Yes	This field should contain random data of random length if present. If absent, the size of this field shall be zero.
See Description	Message Authentication Code (MAC)	Variable	Yes	This field is for illustrative purposes only. The actual location and details of the MAC in the cipher text shall adhere to the AEAD specification for the selected AEAD cipher in `ALGORITHMS` response. AEAD algorithms usually append the MAC to the end of the cipher text.

58 **Table 2 — Attributes Bit Definitions**

Byte	Bit	Field Name	Description
0	[2:0]	LTDtype	This field shall indicate the type of data that is carried in the Layered Transport data field. The values in this field shall be as Table 4 — Supported Layered Transport Data Type defines. See Layered Transport Data Types for additional details.
0	3	LastSegment	If set, the value of this field shall indicate the last segment in a Layered Transport Data Transfer. Otherwise, this field shall be zero.
0	All other bits	Reserved	Reserved
1	All bits	Reserved	Reserved

59 7.1.1 Additional Secured Message Format Details

60 Except for the `MAC`, `LTD Segment`, and `Random Data` fields, all fields are in little endian.

61 The purpose of the random data field is to further obfuscate the application data by hiding its real length. This prevents information from being derived from the real length of the Layered Transport data. In certain scenarios that do not require this obfuscation, the `Random Data` field can have a length of zero.

62 The `Length` field shall be the sum of the length of the following fields:

- `LTD Segment Length` (if present)
- `Random Data` (if present)
- `LTD Segment`
- `MAC`

63 Table 3 — Field presence requirements describes the presence requirement for each field in Secured Messages. Initially, both the Requester and Responder advertise their capabilities through `GET_CAPABILITIES` and `CAPABILITIES` messages. The two capabilities of interest are encryption (`ENCRYPT_CAP`) and message authentication (`MAC_CAP`). For a session to provide message authentication, both the Requester and Responder need to support message authentication. Lastly, for a session to provide both encryption and message authentication, both the Responder and Requester have to support both.

64 The common capabilities between an SPDM Requester and its Responder shall determine the capabilities for all sessions between them. SPDM does not allow a Requester and Responder to support different capabilities for each individual session. In other words, if both the Requester and Responder support both message authentication and encryption, all their sessions shall support message authentication and encryption; not one or the other, but both. Likewise, if there are no common capabilities between the two, Secured Messages shall not be supported.

65 If a session can only provide message authentication, the `Message Authentication Only` column is applicable. If a session provides both, the `Encryption and Message Authentication` column is provided. Only one column applies per session. An encryption-only session shall be prohibited. If no columns apply, this specification is not applicable.

66 In the applicable column, a value of **Present** shall indicate the respective field is present in the Secured Message. A

value of **Absent** shall indicate the respective field shall not be present in Secured Messages. A value of **Conditional** shall indicate the respective field has a presence that is conditional on one or more conditions that are explained at the respective field name(s) in Table 1 — Secured Message field definitions.

67 **Table 3 — Field presence requirements**

Field	Message Authentication Only	Encryption and Message Authentication
Length	Present	Present
Session ID	Present	Present
Sequence Number	See Transport Binding	See Transport Binding
Attributes	Present	Present
Protected Payload Offset	Present	Present
LTD Segment Length	Absent	Present
Random Data	Absent	Present
LTD Segment	Present	Present
MAC	Present	Present
LTD ID	Conditional	Conditional
LTD Segment Number	Conditional	Conditional

68

7.2 Layered Transport Data Types

69 The layered transport data carries different data types depending on the value of the `LTDtype` field in the `Attributes` field as described in Table 1 — Secured Message field definitions. Table 4 — Supported Layered Transport Data Type defines the supported data types. The `LTDtype` field shall contain only values defined in the **Values** column. The **Description** column describes the transport data type that populates into the Layered transport data field for the corresponding `LTDtype`.

70 **Table 4 — Supported Layered Transport Data Type**

Values	Description
0	The Layered transport data shall be the complete Application Data or a segment of it as Application Data defines.
1	The Layered transport data shall be the complete Authorization Record or a segment of it as Authorization Record defines.
2	The Layered transport data shall be the complete Secured Message Error as Secured Message Error defines.

Values	Description
All other values	Reserved.

71 ### 7.2.1 Application Data

72 The transport binding specification is responsible for defining the format and size of the Application data. For some transport bindings, the Application Data will need a specified format to ensure correct processing at the receiver. For example, if the Application Data can carry messages from a range of protocols, the Application Data might need a field indicating which protocol to use for processing the message in the Application Data. If required, such formatting shall be specified by the transport binding specification.

73 ### 7.2.2 Authorization Record

74 When the `LTDtype` is an Authorization record, the format and size of the Layered Transport Data field shall be the Authorization record as DSP0289 defines. Furthermore, in the Authorization record, the size and format of the `MsgToAuthPayload` shall be the same size and format as the Application data.

75 ### 7.2.3 Secured Message Error

76 A Secured Message Error is an error payload sent in response to an error detected by the receiving SPDM endpoint. Table 5 — Secured Message Error format defines the format and size when the `LTDtype` indicates a Secured Message Error, and Table 6 — Secured Message Error Codes defines all the possible error codes and other error information.

77 When `LTDtype` indicates a Secured Message Error, the Layered Transport Data shall be the format and size as Table 5 — Secured Message Error format defines.

78 **Table 5 — Secured Message Error format**

Offset	Field	Length (bytes)	Description
0	ErrorCode	1	This field shall indicate the error code. The possible values in this field shall be as Table 6 — Secured Message Error Codes defines.
1	ExtendedErrorLen	1	This field shall indicate the size of `ExtendedErrorData` field.
2	ExtendedErrorData	`ExtendedErrorLen`	This field, if present, shall have a format and size as Table 6 — Secured Message Error Codes defines for the corresponding `ErrorCode`. Not all error codes have extended error data. If `ExtendedErrorLen` is zero, this field shall be absent.

79 In Table 6 — Secured Message Error Codes, the size in the **Extended Error Length** column shall be used for the size of the `ExtendedErrorData` field in Table 5 — Secured Message Error format for the corresponding error code.

80 **Table 6 — Secured Message Error Codes**

Error Code	Error Name	Extended Error Length (bytes)	Description
0	Reserved	0	Reserved
1	GenTransferError	2	Shall indicate a default huge payload error as Generic Transfer Error defines.
2	InvalidTransfer	2	Shall indicate an invalid transfer as Invalid Transfer Error defines.
3	MissingSegNum	6	Shall indicate a missing segment as Missing Segment Error defines.
4	BufferError	2	Shall indicate an insufficient buffer as Buffer Error defines.
All other values		Reserved	Reserved

81 Lastly, a Secured Message Error shall always be completely transferred with exactly one segment as Layered Transport Data Transfer Mechanism defines.

82 7.2.3.1 Generic Transfer Error

83 A generic transfer error is a default error associated with the Layered Transport Data Transfer Mechanism when no more-specific error codes defined in Table 6 — Secured Message Error Codes apply.

84 This error code contains extended error information. Table 7 — Generic Transfer Extended Error Information defines the format of the extended error information for a transfer error. The extended error info for this error code shall indicate the LTD ID (`LTD ID` field) of the Secured Message that has this error.

85 **Table 7 — Generic Transfer Extended Error Information**

Offset	Field	Length (bytes)	Description
0	LTD ID	2	This field shall indicate the LTD ID of the problematic transfer as Layered Transport Data Transfer Mechanism defines.

86 7.2.3.2 Invalid Transfer Error

87 An invalid transfer occurs when a receiving SPDM endpoint does not receive the first segment with a new LTD ID or when the new transfer exceeds the maximum number of simultaneous LTD transfers (`MaxConcurrentTransfers`) the receiving SPDM endpoint can handle.

88 Since this specification assumes an unreliable transport, the receiving SPDM endpoint should not immediately transmit this error. The receiving SPDM endpoint can wait an appropriate amount of time depending on the underlying transport before considering the first segment to be missing.

89 The extended error info for this error code shall indicate the LTD ID that is invalid. The format of the extended error info shall be as Table 7 — Generic Transfer Extended Error Information defines.

90 #### 7.2.3.3 Missing Segment Error

91 A missing segment error is an error where a segment is missing when transferring an LTD. Since this specification assumes an unreliable transport, the receiving SPDM endpoint should not immediately transmit this error. The receiving SPDM endpoint can wait an appropriate amount of time depending on the underlying transport before considering a segment to be missing.

92 The extended error info for this error shall indicate the LTD ID and LTD segment number that are missing. If more than one segment is missing, then the receiving SPDM endpoint can send this error for each missing segment. The format and size of the extended error info for this error shall be as Table 8 — Missing Segment Extended Error Information defines.

93 **Table 8 — Missing Segment Extended Error Information**

Offset	Field	Length (bytes)	Description
0	LTD ID	2	This field shall indicate the LTD ID of the problematic LTD transfer as Layered Transport Data Transfer Mechanism defines.
2	MissingSegNum	4	This field shall indicate the missing LTD segment number for the transfer associated with the ID in `LTD ID` field.

94 ### 7.2.4 Buffer Error

95 A buffer error is an error where there is not (or there is no longer) sufficient buffer either to receive a new transfer or to continue receiving an active transfer. Upon this error, the receiving SPDM endpoint shall discard all segments associated with the given transfer.

96 The extended error info for this error shall indicate the LTD ID that will be discarded. The format and size of this extended error info shall be as Table 7 — Generic Transfer Extended Error Information defines.

97 ## 7.3 Secured Message protection

98 Secured Messages utilize Authenticated Encryption with Associated Data (AEAD) cipher algorithms in much the same way that TLS 1.3 does. See DSP0274 for an overview of AEAD algorithms.

99 ### 7.3.1 AEAD encryption keys and other secrets

100 SPDM's key schedule produces four major secrets that are used at certain points in the session and each secret is bound to a particular direction of transmission. The encryption keys and initialization vector (IV) are derived from these four major secrets. See Key Schedule in DSP0274 for more details.

101 ## 7.3.2 AEAD requirements

102 This clause discusses the requirements for each parameter to the AEAD functions (`encryption_key` , `nonce` , `associated_data` , `plaintext` and `ciphertext`) depending on the capabilities of the session. See the *Application data* clause in DSP0274 for the AEAD function interfaces and more details. The references below shall be interpreted according to DSP0274 definitions.

103 In general, the MAC covers the associated data and the plain text. Specifically, the MAC covers all fields in Table 1 — Secured Message field definitions. The default length of the MAC shall be 16 bytes for AES-GCM, AEAD_SM4_GCM, and ChaCha20-Poly1305. The transport binding can specify a different MAC length.

104 ### 7.3.2.1 Message Authentication Only session

105 For sessions that are capable of only supporting message authentication, the `associated_data` for AEAD shall be the concatenation of all bytes starting from Session ID to Layered Transport Data inclusively.

106 The fields are as defined in Table 1 — Secured Message field definitions. The text to encrypt, `plaintext` , for AEAD shall be null. Consequently, the text to decrypt, `ciphertext` , shall also be null.

107 ### 7.3.2.2 Encryption and Message Authentication session

108 The `associated_data` for AEAD shall be the concatenation of all bytes starting from `Session ID` through the offset indicated by `Protected Payload Offset` minus 1, inclusively.

109 The fields are as defined in Table 1 — Secured Message field definitions. The text to encrypt, `plaintext` , for AEAD shall be the concatenation of these fields in this order:

1. LTD Segment Length
2. LTD Segment
3. Random Data

110 The fields are as defined in Table 1 — Secured Message field definitions. The text to decrypt, `ciphertext` , shall be the encrypted portion of the Secured Message and the MAC.

111 ## 7.3.3 Per-message nonce derivation

112 The nonce shall never be transmitted in Secured Messages. This means that both the Responder and Requester must internally track the nonce. To ensure proper tracking, the Requester and Responder shall follow the nonce derivation schedule laid out below.

113 Before the creation of the first Secured Message in the session for a given major secret and its derived encryption and IV keys, both the Responder and Requester shall start with an 8-byte sequence number with a value of zero. The sequence number shall be encoded as little endian in memory, so that the least-significant byte is located at address offset `0` and the most-significant byte is located at address offset `7` . For each record, both SPDM endpoints shall follow these steps as prescribed:

1. Retrieve the `iv_length` value according to the selected AEAD cipher suite in the `ALGORITHMS` message.
2. Zero extend the sequence number by appending bytes with a value of `0` from address offset `8` to address offset `iv_length - 1`.
    - The output of this step is called the zero-extended sequence number.
3. Perform a bitwise XOR of the zero-extended sequence number with the appropriate IV derived in the SPDM key schedule.
    - The output of this step is called the per-message nonce.
4. Increment the sequence number by a value of 1 for the next Secured Message.

114 Because different secrets are used for different directions of data transmission, each endpoint would have to track two sequence numbers: one for reception and the other for transmission.

115 Lastly, when a `KEY_UPDATE` occurs, the sequence number shall reset to 0 before sending the first Secured Message using the new session keys.

116 7.3.3.1 Other per-message nonce requirements

117 A Secured Message shall not reuse a sequence number. Furthermore, an SPDM endpoint shall not send Secured Messages out of sequence. The Per-message nonce derivation subclause describes the proper sequence. This requirement does not prevent a transport from sending messages asynchronously or interleave messages as long as the transport supports message interleaving and asynchronous transfers.

118 7.3.4 Encryption requirements

119 A single Secured Message shall contain the complete ciphertext as produced by a single invocation of `AEAD_Encrypt` using the appropriate encryption key for the given direction of transmission, the appropriate per-message nonce, and the selected AEAD Cipher Suite in `ALGORITHMS`. No two or more Secured Messages shall use the same nonce. This requirement does not prevent a transport from sending messages asynchronously or interleave messages as long as the transport supports message interleaving and asynchronous transfers.

120 7.4 Layered Transport Data Transfer Mechanism

121 This specification can transfer any Layered Transport Data (LTD) size up to the maximum supported Layered Transport Data size of the receiving SPDM endpoint. This section details the transfer mechanism for all LTD types. The transport binding could provide an alternative method to transfer LTD types.

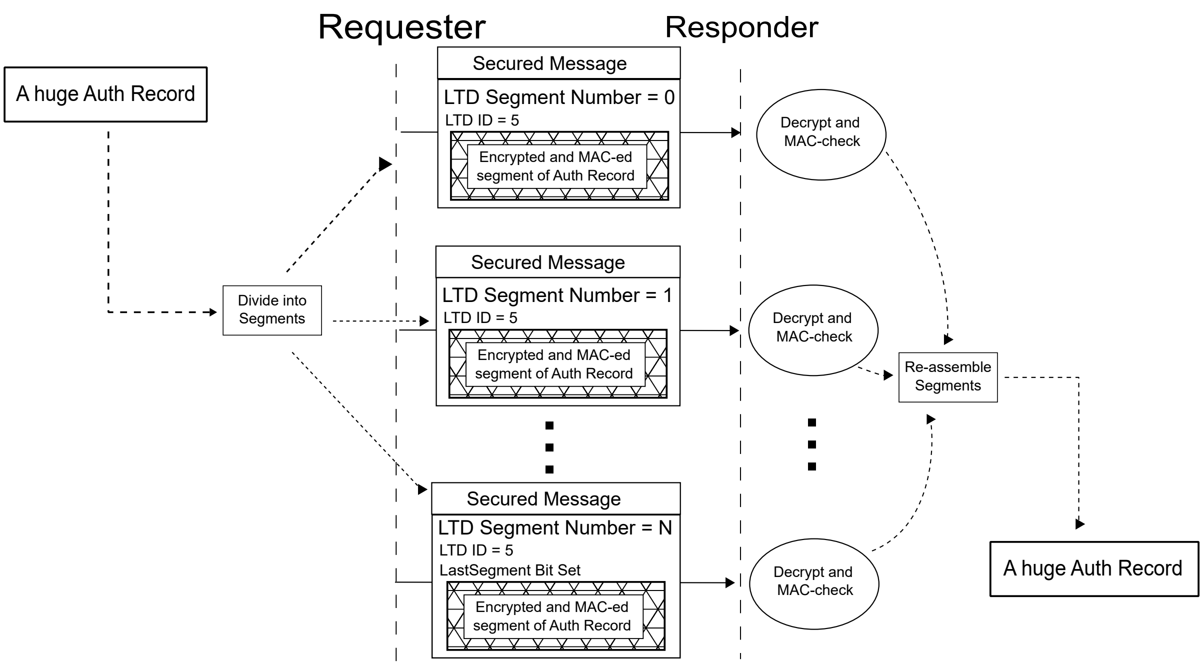
122 This transfer mechanism divides LTD into segments. LTD can be made of one or more segments. The LTD ID identifies all the segments that make up the corresponding LTD, and the segment numbers dictate the order of reassembly. Figure 2 — Huge Auth Record Transfer Example illustrates an example of transferring an Authorization record in multiple segments. In this transfer mechanism, the division of LTD occurs before the encryption and message authentication code (MAC) creation for the sending SPDM endpoint. For the receiving SPDM endpoint, the reassembly of LTD occurs after decryption and message authentication. This allows both SPDM endpoints to correctly and efficiently create or process the Secured Message.

123 A sending SPDM endpoint shall segment an LTD when the size of an LTD exceeds the maximum segment reception buffer of the receiving SPDM endpoint. There can be other reasons a sending SPDM endpoint segments an LTD.

124 All receiving SPDM endpoints shall populate the Buffer Parameters data type as Table 16 — Buffer Parameters data defines. If a receiving SPDM endpoint does not populate this data type as Buffer parameters data defines, the secured session shall terminate unless the transport binding provides an alternative transfer mechanism.

125 Note that Figure 2 — Huge Auth Record Transfer Example does not show all fields necessary in a Secured Message.

126



127 **Figure 2 — Huge Auth Record Transfer Example**

128 A sending SPDM endpoint can send multiple LTDs with different LTD IDs at the same time. The sending SPDM endpoint shall never send messages that exceed the maximum simultaneous payload transfers (`MaxConcurrentTransfers`) of the receiving SPDM endpoint. The LTD ID shall uniquely identify each LTD transfer. An LTD ID should increment by one with each payload transfer. An LTD ID value can be reused if it does not overlap with an active LTD transfer.

129 The LTD transfer mechanism shall be bound to the session in which the transfer occurs. This means an LTD transfer in one session can occur simultaneously with an unrelated LTD transfer in another session. Thus, the requirements defined in this section are bound within the session.

130 Segments shall be numbered and shall be transferred in order. The segments and their order of transfer are described as follows:

- The first segment shall be assigned a numeric value of 0, the second segment shall be assigned a numeric value of 1, the third segment shall be assigned a numeric value of 2, and this pattern shall continue up to and including the last segment. Each of these numeric values is called a segment number.
- The first segment shall contain the first set of bytes of the huge payload, the second segment shall contain the

second set of bytes, the third segment shall contain the third set of bytes, and this pattern shall continue up to and including the last segment.

- All segments of the same LTD ID shall represent all bytes of the LTD without altering the message in any way.
- The order of transfer shall start with segment number 0 and shall continue with sequentially increasing segment numbers up to and including the last segment.
- All segments should utilize the maximum allowed segment size for efficient transfer.
- The last segment shall have the [`Secured Message`].`Attributes`/`LastSegment` bit set. All other segments shall not set this bit.
- The [`Secured Message`].`Attributes`/`LTDtype` field shall be the same for all segments.

131 When an LTD is transferred completely using a single segment, the single segment is both the first and last segment and has a segment number of zero.

132 7.4.1 Additional Considerations for LTD Transfers

133 For reasons of simplicity, expediency, and efficiency, this layer does not provide reliability requirements. This specification assumes either the underlying transport or the payload protocol is responsible for reliability.

134 The receiving SPDM endpoint should send a Secured Message Error when it encounters an LTD transfer error. Alternatively, the receiving SPDM endpoint can discard the invalid LTD transfer entirely and silently. The transport binding specification can offer additional error-handling mechanisms for unsuccessful transfers. Higher-layer protocols can also define additional error-handling mechanisms such as retries to ensure a payload gets transferred (if such mechanisms are not already defined).

135 Buffer sizes should reflect a size that prevents higher-layer protocols or Application data from needing another similar transfer mechanism to maximize the efficiency of transfers. For example, the SPDM `MaxSPDMmsgSize` should be at least the size of [`Buffer Parameters data`].`MaxLTDsize`.

136 7.4.2 LTD transfer timing

137 This section applies to payloads sent using more than one segment.

138 In the absence of a transport-defined timeout mechanism, the receiving SPDM endpoint shall wait at least 2 seconds from the completion of one segment of a multiple-segment transfer to the receipt of another segment of the same transfer. If no additional segment of the transfer is received within this timeout period, the receiving endpoint can discard all previously received segments of the incomplete transfer. The timeout period is measured from the end of the last received segment to the start of the next segment. The receipt of any other messages does not reset or extend the timeout period.

139 7.4.3 LTD retries

140 If the sender of a payload transfer detects an error in transmission of a segment, the sender can retry sending the segment. The retry shall be identical to the first transmission attempt, including the same segment number. If the retry is different from the first message, the receiver can silently discard the message or reject it with a Secured

Message Error response with `ErrorCode` = `GenTransferError`. If the payload is sent using more than one segment, the receipt of a retry shall reset the timeout period as LTD transfer timing describes.

141 # 8 Version support

142 To advertise the supported Secured Message version of an SPDM Requester for a particular transport, Table 9 — Secured Message transport binding version format defines the fields necessary to specify the transport binding specification version of Secured Messages.

143 **Table 9 — Secured Message transport binding version format**

Bit	Field	Value
[15:12]	MajorVersion	Shall be the major version of the transport binding. See DSP0274 for the description of a major version.
[11:8]	MinorVersion	Shall be the minor version of the transport binding. See DSP0274 for the description of a minor version.
[7:4]	UpdateVersionNumber	Shall be the update version of the transport binding. See DSP0274 for the description of an update version.
[3:0]	Alpha	Shall be the alpha version of the transport binding. For released versions, this field shall be zero. See DSP0274 for the description of an alpha version.

144 The transport binding specification of Secured Messages may offer a more descriptive or definitive statement on compatibility between major, minor, and update versions.

145 Table 10 — Secured Message transport binding version list format describes a format to list all supported versions of the Secured Message transport binding.

146 **Table 10 — Secured Message transport binding version list format**

Offset	Field	Size (in bytes)	Description
0	VersionCount	1	Shall indicate the total number (T) of Secured Message transport binding versions listed in `VersionsList`
1	VersionsList	T * 2	Shall list all versions that are supported by an SPDM Requester for the given transport. The format for this field shall be the format described by Table 9 — Secured Message transport binding version format.

147 The transport binding specification shall define which version(s) of DSP0277 (this specification) it binds to. DSP0277 recommends binding to compatible versions of DSP0277 for a given version of the transport binding specification.

148 Additionally, the transport binding specification should define the values to populate in `VersionsList`. If the transport binding specification does not define the values to populate in `VersionsList`, then the version of the transport binding specification, itself, shall be the value to populate in `VersionsList`. For example, if version 1.3.4 of a transport binding specification states that it binds to version 1.0.0 of DSP0277 and does not define the values to populate in `VersionsList`, then the value used for `VersionsList` is 1.3.4. In another example, a 2.4 version of the transport binding specifically defines a value of 4.5 as the value to use for `VersionsList`. The Requester should populate `VersionsList` with all versions it supports.

149 8.1 Version selection

150 Version discovery and selection occur during Session-Secrets-Exchange. First, the SPDM Requester shall advertise its list of supported versions through the `OpaqueData` field of a Session-Secrets-Exchange request. The Requester shall use the Secured Message opaque element as Table 11 — Secured Message opaque element defines to specify its list in Supported version list data.

151 Lastly, the Responder shall select a version among the ones that are supported by the Requester and communicate the selected version in the `OpaqueData` field in the Session-Secrets-Exchange response. The Responder shall use Table 11 — Secured Message opaque element to specify its selected version in Version selection data. The Responder should select the latest supported common version. From that point on, both the SPDM Requester and Responder shall not change this version for that session. If the Responder cannot select a version, an ERROR response shall be sent with `ErrorCode=InvalidRequest`.

152 # 9 Transport requirements or allowances

153 This clause and subclauses describe various requirements or flexibility allowed at the transport layer.

154 ## 9.1 Transmission reliability

155 Secured Messages rely on the transport to perform reliable lossless delivery. The transport defines the mechanisms to ensure the transmission of data, which can include retries. Furthermore, this specification expects the transport to either deliver Secured Messages in order or allow the receiver to determine the correct order of transmission of Secured Messages. In the event that transmission or reception fails, an SPDM Requester or Responder may terminate the session or restart a new one.

156 If a transport cannot provide the aforementioned characteristics, the transport binding may add the sequence number as described in Per-message nonce derivation as a field to Secured Message and provide additional guidance, if any, to properly utilize the sequence number to authenticate or decrypt the Secured Message. See DTLS 1.3 for further guidance.

157 ## 9.2 Certain SPDM message allowances

158 If possible, the transport binding specification should take full advantage of asynchronous and bidirectional communication to allow messages such as `KEY_UPDATE` and `HEARTBEAT` to be sent directly from a Responder without any other assistance such as a sideband alerting mechanism or SPDM's `GET_ENCAPSULATED_REQUEST` mechanism. The transport binding specification shall address this.

159 ## 9.3 ERROR response message allowances

160 Furthermore, the `ERROR` message may be sent without an SPDM request when the error code is a decryption error (`ErrorCode=DecryptError`) to indicate that the Secured Message that was received could not be decrypted or authenticated properly. In addition, both the SPDM Requester and Responder may send an `ERROR` message with `ErrorCode=DecryptError` . This is especially useful for data sent at the application layer. In other words, in this scenario, the `ERROR` response message is behaving as a response to the inability to decrypt or authenticate the received Secured Message. In the event an SPDM endpoint receives this particular error message, the SPDM endpoint should terminate the session.

161 ## 9.4 Key update allowances

162 Unless otherwise specified by the transport, in a secure session, the SPDM endpoints can continue to transfer Secured Messages while a key update using the `KEY_UPDATE` message is in progress.

163 On the sender end, after the sender sends out a `KEY_UPDATE` (Key Operation == `UpdateKey`) request to the receiver

and before the sender receives a corresponding `KEY_UPDATE_ACK` response from the receiver, the sender uses the current (old) session key for protecting outgoing Secured Messages and retries of `KEY_UPDATE` (Key Operation == `UpdateKey`). After the sender receives the `KEY_UPDATE_ACK` response from the receiver, the sender deletes the old session key and starts using the new session key for protecting outgoing `KEY_UPDATE` (Key Operation == `VerifyNewKey`) request and Secured Messages. The first message that is protected by the new session key is the outgoing `KEY_UPDATE` (Key Operation == `VerifyNewKey`) request.

164 On the receiver end, the receiver keeps the old session key after sending out the `KEY_UPDATE_ACK` (Key Operation == `UpdateKey`) response, because the receiver may still receive messages protected by the old session key (either Secured Messages or retries of `KEY_UPDATE` (Key Operation == `UpdateKey`)). Those messages were sent by the sender before `KEY_UPDATE_ACK` (Key Operation == `UpdateKey`) was processed by the sender. In other words, after sending out the `KEY_UPDATE_ACK` (Key Operation == `UpdateKey`) response, the receiver may have to try decrypting an incoming message twice, with the new session key and the old session key, respectively. If the transport guarantees the order of message delivery, then the receiver deletes the old session key as soon as it receives a message protected by the new session key. If the transport does not guarantee the order of message delivery, then the earliest the receiver can delete the old session key is when it receives a message using the new session key and the longest it can keep the old session key is when all prior messages have been decrypted or messages encrypted with the old key can be considered lost. These situations should be addressed by the transport binding specification.

165 9.5 AEAD Limit

166 The Secured Message in an SPDM session uses an 8-byte sequence number to derive AEAD IV. The maximum sequence number causing the session termination by an endpoint is known as the AEAD limit. By default, the AEAD limit is 2^{64}. An endpoint may use a smaller AEAD limit. The Requester and the Responder can exchange their own AEAD limit information via Table 15 — AEAD limit data format. The endpoint may send `KEY_UPDATE` to keep the session alive before the sequence number reaches the AEAD limit.

167
10 Secured Messages opaque data

168 In many SPDM requests and responses, an opaque data field exists to accommodate use cases for transports, standards organizations, and vendors. SPDM defines a General opaque data format (`OpaqueDataFmt1` is selected). The Requester and Responder shall support `OpaqueDataFmt1` , and the Responder shall select `OpaqueDataFmt1` in the `ALGORITHMS` message. All Secured Message opaque data shall be transferred over Session-Secrets-Exchange messages.

169 Secured Messages opaque data shall use the General opaque data format. Secured Messages shall use an `ID` of zero to indicate DMTF with a `VendorLen` of zero in the General opaque data format. The size and format of the `OpaqueElementData` , as SPDM defines, shall be the size and format of the Secured Message opaque format.

170 Note: The Secured Message opaque data is the same as version 1 of this specification for SPDM 1.2 or later.

171
10.1 Secured Message opaque data format

172 The Secured Message opaque element data is the general format for all Secured Message opaque data. Table 11 — Secured Message opaque element describes the implementation.

173 **Table 11 — Secured Message opaque element**

Offset	Field	Length (bytes)	Description
0	SMDataVersion	1	Shall identify the format of the remaining bytes. The value shall be 1.
1	SMDataID	1	Shall be the identifier for the Secured Message data type. Later clauses of this specification describe the allowed values.
2	SMData	SMDataLen	Shall be the data corresponding to `SMDataID` . Later clauses of this specification describe this format.

174 Table 12 — Secured Message Data Types lists all the possible data types. The values in the **SM Data ID Value** column shall be the allowed values for the `SMDataID` field.

175 **Table 12 — Secured Message Data Types**

SM Data Type	SM Data ID Value	Description
VersionSelectionOE	0	Shall be the Version selection data type.
SupportedVersionListOE	1	Shall be the Supported version list data type.
AEADlimitOE	2	Shall be the AEAD limit data type.
BufferParametersOE	16	Shall be the Buffer Parameters data type.
Reserved	All other Values	Reserved

176 10.1.1 Version selection data

177 The Version selection data shall be used by the SPDM responder to communicate the selected Secured Message version. This data type shall be allowed only in a Session-Secrets-Exchange Response. The Version selection data shall populate the `SMData` field of Table 11 — Secured Message opaque element. An SPDM Responder populates this information. The `SMDataID` shall be the value for `VersionSelectionOE`. The size and format of the `SMData` field shall be the size and format as Table 13 — Secured Message version selection data format defines.

178 **Table 13 — Secured Message version selection data format**

Offset	Field	Length (bytes)	Description
0	SelectedVersion	2	Shall be the selected Secured Message Version. See Table 9 — Secured Message transport binding version format for the format of this field.

179 10.1.2 Supported version list data

180 The Supported version list data shall be used by the Requester to communicate its list of supported Secured Message transport binding versions. The Supported version list data shall populate the `SMData` field of Table 11 — Secured Message opaque element. This data type shall be allowed only in a Session-Secrets-Exchange Request. An SPDM Requester populates this information. The `SMDataID` shall be the value for `SupportedVersionListOE`. The size and format of the `SMData` field shall be the size and format as Table 14 — Supported version list data format defines.

181 **Table 14 — Supported version list data format**

Offset	Field	Length (bytes)	Description
0	SecuredMsgVers	(`T` * 2) + 1	Shall be the format described in Table 10 — Secured Message transport binding version list format.

182 10.1.3 AEAD limit data

183 The AEAD limit data shall populate the `SMData` field of Table 11 — Secured Message opaque element to communicate the Requester or Responder selected AEAD limit. This data type shall be allowed only in a Session-Secrets-Exchange Request. An SPDM Requester or Responder populates this information. The `SMDataID` shall be the value for `AEADlimitOE` . The size and format of the `SMData` field shall be the size and format as Table 15 — AEAD limit data format.

184 **Table 15 — AEAD limit data format**

Offset	Field	Length (bytes)	Description
0	AeadLimitExponent	1	Shall be an exponent of base 2 that is used to calculate `AeadLimit`. The value shall be less than or equal to 64. The equation for `AeadLimit` shall be $2^{AeadLimitExponent}$. The Requester sets this value to inform the responder that the Requester will terminate a session when the session sequence number reaches the `AeadLimit` of the Requester. If this table is present, the Requester shall interpret the `AeadLimitExponent` from the Responder, and the Requester should use `KEY_UPDATE` to keep the session alive before the session sequence number reaches the `AeadLimit` of the Requester or the `AeadLimit` of the Responder. The Responder sets this value to inform the requester that the Responder will terminate a session when the session sequence number reaches the `AeadLimit` of the Responder. If this table is missing, the `AeadLimitExponent` shall be interpreted as 64 by the Requester.

185 ### 10.1.4 Buffer Parameters data

186 The buffer parameters data type indicates the size of the buffers for receiving Secured Messages. The `SMDataID` shall be the value for `BufferParametersOE`. The size and format of the `SMData` field shall be as Table 16 — Buffer Parameters data defines. A receiving SPDM endpoint communicates these parameters to the sending SPDM endpoint for proper reception of LTD as Layered Transport Data Transfer Mechanism defines.

187 This data type shall only populate in Session-Secrets-Exchange messages.

188 **Table 16 — Buffer Parameters data**

Offset	Field	Length (bytes)	Description
0	MaxSegmentSize	4	Shall be the maximum size of an LTD segment the receiving SPDM endpoint can receive. The sending SPDM endpoint shall not send a Secured Message with the `[Secured Message]`.`LTD Segment Length` exceeding this value. The value of this field shall be greater than the maximum possible size of all `LTDtype`s that must be sent in a single segment as Layered Transport Data Transfer Mechanism defines. For this version of this specification, this value shall be greater than the maximum possible value of the Secured Message Error which is 257 bytes.
4	MaxLTDsize	4	Shall be the maximum size of a Layered Transport Data that the receiving SPDM endpoint can receive. The sending SPDM endpoint shall not transfer an LTD whose size is greater than the value of this field (that is, the sum of all segment sizes for the corresponding LTD). The value of this field shall be equal to or greater than the `MaxSegmentSize`.
8	MaxConcurrentTransfers	4	This field shall indicate the maximum number of simultaneous LTD transfers that the receiving SPDM endpoint can handle at any time. The value of this field shall be greater than zero.

189

11 ANNEX A (informative) Sequence number layout example

190 In this example, the 8-byte sequence number with value `0x01FF02EE03DD04CC`, located at address offsets `0x0` through `0x7`, is zero-extended for the Per-message nonce derivation. `iv_length` is `12`, and address offsets `0x8` through `0xB` contain the extended zeros.

Address Offset	Value
0x0	0xCC
0x1	0x04
0x2	0xDD
0x3	0x03
0x4	0xEE
0x5	0x02
0x6	0xFF
0x7	0x01
0x8	0x00
0x9	0x00
0xA	0x00
0xB	0x00

191 12 ANNEX B (informative) Change log

192 12.1 Version 2.0.0 (2025-10-30)

- Initial release

193 13 Bibliography

194 DMTF DSP4014, *DMTF Process for Working Bodies*,
https://www.dmtf.org/dsp/DSP4014