1 **Document Identifier: DSP0287**

2 **Date: 2024-01-17**

3 **Version: 1.0.0WIP99**

4 # SPDM over TCP Binding Specification

Information for Work-in-Progress version:

5 **IMPORTANT:** This document is not a standard. It does not necessarily reflect the views of DMTF or its members. Because this document is a Work in Progress, this document may still change, perhaps profoundly and without notice. This document is available for public review and comment until superseded.

6 **Provide any comments through the DMTF Feedback Portal:** https://www.dmtf.org/standards/feedback

7 **Supersedes: None**

8 **Document Class: Normative**

9 **Document Status: Work in Progress**

10 **Document Language: en-US**

11 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, 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.

12 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 product, protocols, or testing procedures. DMTF shall have no liability to any party implementing such standard, 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patent owner or claimant, and shall have no liability or responsibility for costs or losses incurred if a standard is withdrawn or modified after publication, and shall be indemnified and held harmless by any party implementing the standard from any and all claims of infringement by a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.

13 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.

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

CONTENTS

16 # 1 Foreword

17 The Security Protocols and Data Models (SPDM) Working Group prepared the *SPDM over TCP Binding Specification* (DSP0287).

18 DMTF is a not-for-profit association of industry members that promotes enterprise and systems management and interoperability. For information about the DMTF, see DMTF.

19 ## 1.1 Acknowledgments

20 The DMTF acknowledges these individuals' contributions to this document:

21 **Contributors:**

- Steven Bellock — NVIDIA Corporation
- James Borden — Kioxia
- Eduardo Cabre — Intel Corporation
- Daniil Egranov — Arm Limited
- Brett Henning — Broadcom Inc.
- Jeff Hilland — Hewlett Packard Enterprise
- Guerney Hunt — IBM
- Raghupathy Krishnamurthy — NVIDIA Corporation
- Eliel Louzoun — Intel Corporation
- Chandra Nelogal — Dell Technologies
- Alexander Novitskiy — Intel Corporation
- Jim Panian — Qualcomm Inc.
- Scott Phuong — Axiado Corporation
- Xiaoyu Ruan — Intel Corporation
- Jiewen Yao — Intel Corporation

22 # 2 Introduction

23 The SPDM specification is a standard published by DMTF. The SPDM defines messages, data objects, and sequences for performing message exchanges between two entities. The message exchanges include authentication, provisioning of identities, measurements, session key exchange protocols, and other related capabilities. The SPDM message exchanges can be performed over various transport mechanisms. This document specifies binding SPDM messages to *Transmission Control Protocol (TCP)*.

24 ## 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.

25 # 3 Scope

26 This document binds Out-of-Session SPDM messages and In-Session SPDM messages to TCP and further defines the transport specific details.

27 ## 3.1 Normative references

28 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.

- IETF RFC 9293, "Transmission Control Protocol (TCP)", https://datatracker.ietf.org/doc/html/rfc9293
- DMTF DSP0274, *Security Protocol and Data Model (SPDM) Base Specification any version*, https://www.dmtf.org/dsp/DSP0274
- DMTF DSP0277, *Secured Messages using SPDM Specification any version*, https://www.dmtf.org/dsp/DSP0277
- IANA, *TCP port number listing*, https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml?search=4194
- *ISO/IEC Directives, Part 2, Principles and rules for the structure and drafting of ISO and IEC documents*, https://isotc.iso.org/livelink/livelink.exe?func=ll&objId=4230456&objAction=browse&sort=subtype

29 ## 3.2 Terms and definitions

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

31 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.

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

33 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.

34 The terms that DSP0277 and DSP0274 define also apply to this document.

35 This specification uses these terms:

Term	Definition
TCP connection	A transport link that is set up using the TCP protocol between an SPDM Requester and an SPDM Responder. “TCP connection” is also referred to as “connection” in this specification.
SPDM session	A channel that is set up using the SPDM protocol with message authentication and with or without message encryption for communicating data over TCP. “SPDM session” is also referred to as “session” in this specification.
reach out model	The SPDM Responder initiates the TCP connection with the SPDM Requester.
reach down model	The SPDM Requester initiates the TCP connection with the SPDM Responder.
reference measurement	A baseline or known-good (“golden”) measurement to be compared against an SPDM responder’s measurement reported in the `MEASUREMENTS` response.
verification policy	A set of rules that direct the evaluation of the an SPDM responder’s measurement reported in the `MEASUREMENTS` response against the reference measurement. “Verification policy” is also referred to as “policy” in this specification.

3.3 Symbols and abbreviated terms

The abbreviations or notations defined in DSP0277 and DSP0274 apply to this document.

3.4 Binding Information

This version of this specification binds to *Security Protocol and Data Model (SPDM)* specification (DSP0274).

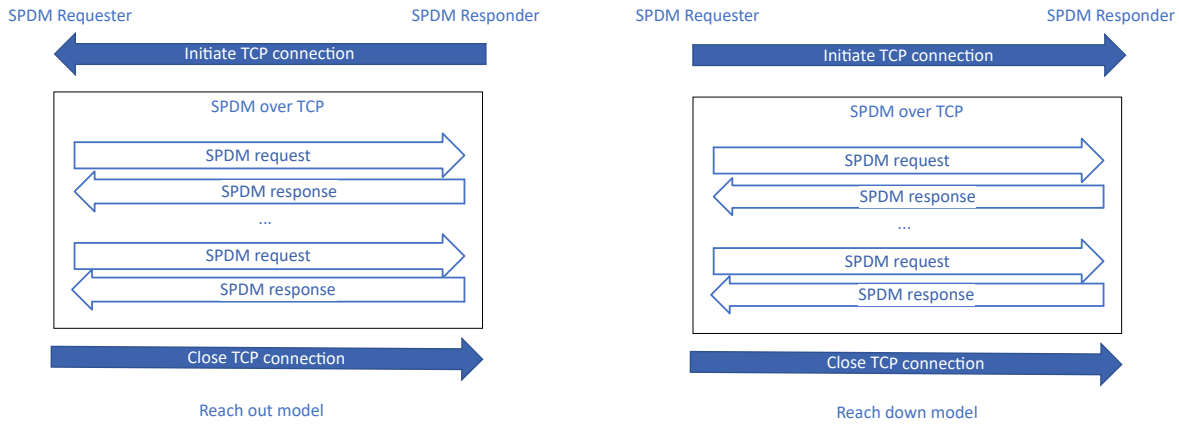
This version of this specification binds to *Secured Messages using SPDM* specification (DSP0277).

41 # 4 Overview

42 This specification describes transmitting SPDM messages over TCP transport between two endpoints, the SPDM Requester (also referred to as Requester) and the SPDM Responder (also referred to as Responder). The diagram on the left of Figure 1 shows the reach out model where the Responder initiates the TCP connection. The diagram on the right of Figure 1 shows the reach down model where the Requester initiates the TCP connection.

43 **Figure 1 — SPDM over TCP**

44



45 When the Requester finishes sending SPDM requests and processing SPDM responses with the Responder, the Requester should close the TCP connection. The Responder should not close the TCP connection before the Requester, unless errors or timeouts have occurred.

46 This specification covers the following topics:

1. Allocate a TCP port number for SPDM messages.
2. Define TCP data section formats for SPDM messages.
3. Define `VENDOR_DEFINED_REQUEST` and `VENDOR_DEFINED_RESPONSE` messages for use cases specific to SPDM over TCP.

47 The SPDM messages may be within an SPDM session or outside of an SPDM session.

48 When SPDM is used to protect the data transported over TCP, other protection mechanisms (such as Transport Layer Security (TLS)) shall not be used.

5 TCP Port Number and TCP Connection

The IANA has assigned TCP port number **4194** for SPDM.

The TCP connection listener shall bind to and listen at port 4194. In the TCP header (Figure 1 of RFC 9293), the TCP connection initiator shall initiate a TCP connection with the TCP connection listener by setting “Destination Port” to 4194. The TCP connection initiator may choose its own port (“Source Port”) for the connection. If a TCP connection initiator initiates concurrent TCP connections with multiple TCP connection listeners, the TCP connection initiator shall use different Source Ports with different TCP connection listeners.

An SPDM communication between two endpoints shall not span or multiplex over multiple TCP connections. When a TCP connection is terminated, the SPDM communication(s) relied on the TCP connection shall also be terminated.

6 TCP Data Section Format for SPDM Messages

The “Data” section in the TCP header shall contain a 4-byte *TCP SPDM binding header* followed by the *SPDM message payload*. There are two types of SPDM message payload: Out-of-Session Message and In-Session Message.

6.1 Header

Table 1 defines the structure of the TCP SPDM binding header.

Table 1 — TCP SPDM binding header data structure

Byte offset	Field	Size (bytes)	Description
0	PayloadLen	2	Shall be the total length of the `BindingVer` field (1 byte), `MessageType` field (1 byte), and the SPDM message payload.
2	BindingVer	1	Shall be 0x01 for this version of the binding specification.
3	MessageType	1	Shall indicate the SPDM message payload type. - 0x05: Out-of-Session Message. - 0x06: In-Session Message. - other values: reserved.

6.2 Out-of-Session Message

An “Out-of-Session Message” refers to an SPDM message that is not protected by a session key negotiated from `KEY_EXCHANGE` or `PSK_EXCHANGE`. The TCP data section for an Out-of-Session Message uses the format shown in Figure 2 and Table 2.

Figure 2 — TCP data section format of Out-of-Session Message

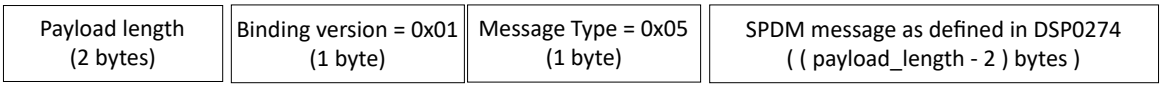


Table 2 — TCP data section data model of Out-of-Session Message

Byte offset	Field	Size (bytes)	Description
0	BindingHeader	4	Shall be TCP SPDM binding header as defined in Table 1.

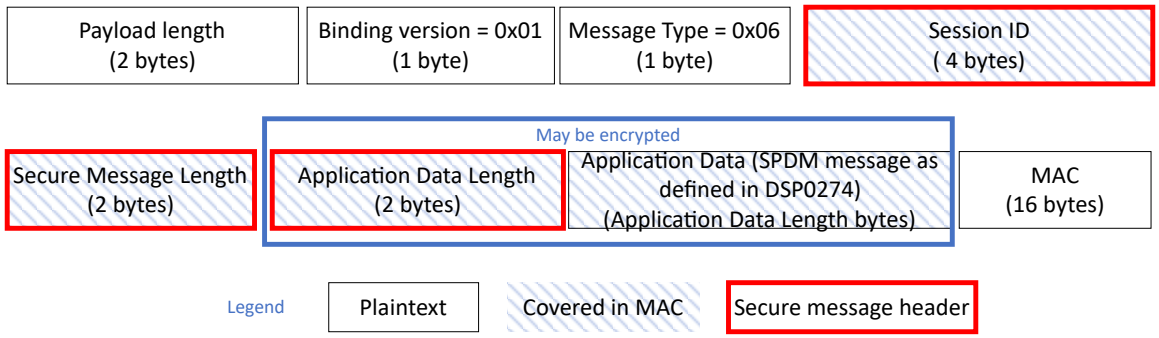
Byte offset	Field	Size (bytes)	Description
4	SpdmMessage	varies	Shall be an SPDM message as defined in DSP0274, including the SPDM header. This SPDM message shall be allowed outside of a session, as specified in Table “SPDM request and response messages validity” of DSP0274.

63 ## 6.3 In-Session Message

64 An “In-Session Message” refers to an SPDM message that is protected by a session key negotiated from `KEY_EXCHANGE` or `PSK_EXCHANGE`. The TCP data section for an In-Session Message uses the format shown in Figure 3 and Table 3.

65 **Figure 3 — TCP data section format of In-Session Message**

66



67 **Table 3 — TCP data section data model of In-Session Message**

Byte offset	Field	Size (bytes)	Description
0	BindingHeader	4	Shall be TCP SPDM binding header as defined in Table 1.
4	SecuredMessage	varies	Shall follow the format defined in Table “Secured Message fields definition” of DSP0277. Specifically: - The “Random Data” field shall be absent. - The “Partial Sequence Number” field shall be absent. - The “Application Data” field shall be an encrypted and MAC’ed or MAC’ed SPDM message as defined in DSP0274, including the SPDM header. This SPDM message shall be allowed in a session, as specified in Table “SPDM request and response messages validity” of DSP0274.

68

7 VENDOR_DEFINED_REQUEST and VENDOR_DEFINED_RESPONSE messages for SPDM over TCP

69 For Out-of-Session and In-Session Message formats, the SPDM message payload may contain a “native” SPDM message or a vendor-defined SPDM message.

- DSP0274 defines data models for native request and response messages, such as `GET_VERSION`, `GET_MEASUREMENTS`, `KEY_EXCHANGE`, and so on.
- Individual binding specifications define data models for `VENDOR_DEFINED_REQUEST` (request message code 0xFE) and `VENDOR_DEFINED_RESPONSE` (responder message code 0x7E) and their flows. This specification defines `VENDOR_DEFINED_REQUEST` and `VENDOR_DEFINED_RESPONSE` messages for use cases of SPDM over TCP binding.

70 Table 4 defines the data model of `VENDOR_DEFINED_REQUEST` with field values specific to SPDM over TCP binding.

71 **Table 4 — Data model of `VENDOR_DEFINED_REQUEST` for SPDM over TCP**

Byte offset	Field	Size (bytes)	Description
0	SPDMVersion	1	Shall be the the SPDM version negotiated in `GET_VERSION` / `VERSION`.
1	RequestResponseCode	1	Shall be 0xFE.
2	Param1	1	Reserved.
3	Param2	1	Reserved.
4	StandardID	2	Shall be 0x0000 (DMTF).
6	Len	1	Shall be 0x00 (`VendorID` is absent for standards-defined `VendorDefinedReqPayload`).
7	ReqLength	2	Shall be the length of `VendorDefinedReqPayload`.
9	VendorDefinedReqPayload	`ReqLength`	Shall follow the format defined in Table 5.

72 As specified in DSP0274, binding specifications published by the DMTF (where StandardID is 0x0000) shall define the `VendorDefinedReqPayload` field in `VENDOR_DEFINED_REQUEST` and the `VendorDefinedRespPayload` field in `VENDOR_DEFINED_RESPONSE`. For SPDM over TCP, data models of `VendorDefinedReqPayload` and `VendorDefinedRespPayload` are defined in Table 5 and Table 7, respectively.

73 **Table 5 — Data model of `VendorDefinedReqPayload` for SPDM over TCP**

Byte offset	Field	Size (bytes)	Description
0	TCP_DSPNumber	2	Shall indicate this SPDM over TCP specification's DSP number as a 16-bit integer, 0x011F.
2	TCP_DSPVersion	2	Shall be the version number of this SPDM over TCP specification (DSP0287). For this version, - MajorVersion = 1. - MinorVersion = 0. - UpdateVersionNumber = 0. - Alpha = 0.
4	TCP_SubRequestID	1	Shall indicate the vendor-defined request specified in this specification. - 0x00: GET_SERVICE_REQUEST. - 0x02: VERIFICATION_RESULTS. - 0x05: SET_REFERENCE. - 0x06: SET_POLICY. - other values: reserved.
5	TCP_ReqPayloadLen	2	Shall be the length of `TCP_ReqPayload`.
7	Reserved	1	Reserved.
8	TCP_ReqPayload	`TCP_ReqPayloadLen`	Defined by individual vendor-defined requests specified in this specification.

74 Table 6 defines the data model of `VENDOR_DEFINED_RESPONSE` with field values specific to SPDM over TCP binding.

75 **Table 6 — Data model of `VENDOR_DEFINED_RESPONSE` for SPDM over TCP**

Byte offset	Field	Size (bytes)	Description
0	SPDMVersion	1	Shall be the the SPDM version negotiated in `GET_VERSION` / `VERSION`.
1	RequestResponseCode	1	Shall be 0x7E.
2	Param1	1	Reserved.
3	Param2	1	Reserved.
4	StandardID	2	Shall be 0x0000 (DMTF).
6	Len	1	Shall be 0x00 (`VendorID` is absent for standards-defined `VendorDefinedRespPayload`).
7	RespLength	2	Shall be the length of `VendorDefinedRespPayload`.
9	VendorDefinedRespPayload	`RespLength`	Shall follow the format defined in Table 7.

76 **Table 7 — Data model of `VendorDefinedRespPayload` for SPDM over TCP**

Byte offset	Field	Size (bytes)	Description
0	TCP_DSPNumber	2	Shall indicate this SPDM over TCP specification's DSP number as a 16-bit integer, 0x011F.
2	TCP_DSPVersion	2	Shall be the version number of this SPDM over TCP specification (DSP0287). For this version, - MajorVersion = 1. - MinorVersion = 0. - UpdateVersionNumber = 0. - Alpha = 0.
4	TCP_SubResponseID	1	Shall indicate the vendor-defined response specified in this specification. - 0x80: SERVICE_REQUEST. - 0x82: VERIFICATION_RESULTS_ACK. - 0x85: SET_REFERENCE_ACK. - 0x86: SET_POLICY_ACK. - other values: reserved.
5	TCP_RespPayloadLen	2	Shall be the length of `TCP_RespPayload`.
7	Reserved	1	Reserved.
8	TCP_RespPayload	`TCP_RespPayloadLen`	Defined by individual vendor-defined responses specified in this specification.

77 The vendor-defined messages' validity with regard to session is summarized in Table 8.

78 **Table 8 — Validity of Vendor-Defined Messages for SPDM over TCP**

Request	Response	Outside of a session	Session (application phase)
GET_SERVICE_REQUEST	SERVICE_REQUEST	Allowed	Allowed
VERIFICATION_RESULTS	VERIFICATION_RESULTS_ACK	Prohibited	Allowed, mutual authentication required

Request	Response	Outside of a session	Session (application phase)
SET_REFERENCE	SET_REFERENCE_ACK	Allowed only if data structure of the reference measurements guarantees integrity, authenticity, and freshness	Allowed; Mutual authentication required, if data structure of the reference measurements does not guarantee integrity, authenticity, or freshness
SET_POLICY	SET_POLICY_ACK	Allowed only if data structure of the policy guarantees integrity, authenticity, and freshness	Allowed; Mutual authentication required, if data structure of the policy does not guarantee integrity, authenticity, or freshness

79

7.1 VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST) and VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)

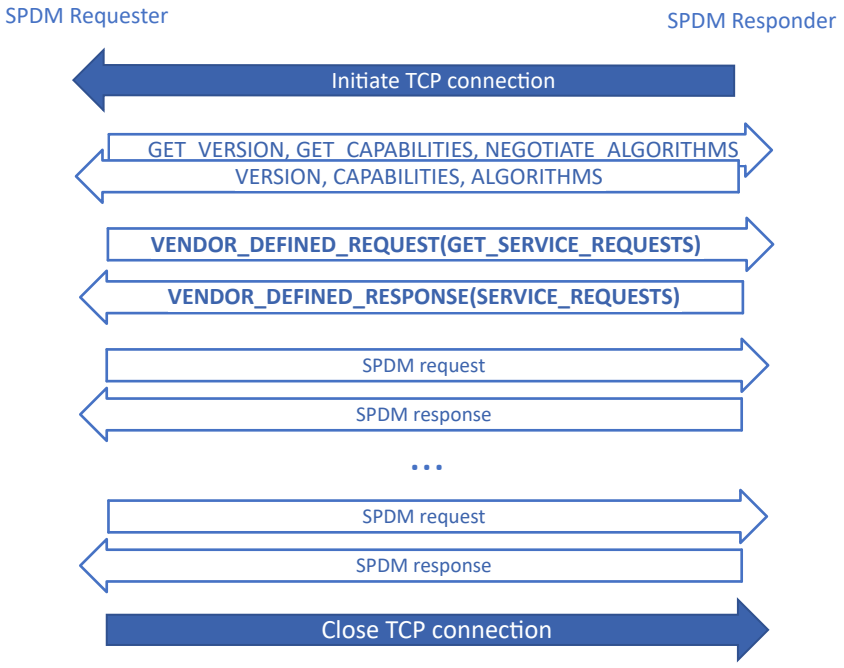
80 In the reach out model (left of Figure 1), the Requester may need to inquire about the Responder's reason for initiating the TCP connection. To do this, the Requester may send `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)` after receiving the `CAPABILITIES` or after establishing a secure session. If the Requester and the Responder have previously cached a negotiated state, then the Requester may send `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)` as the first SPDM message. The `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)` and `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)` messages may be sent within a session or outside of a session.

81 If a Requester or Responder supports SPDM over TCP with the reach out model, then it should support `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)` and `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)`. If a Requester or Responder supports SPDM over TCP with the reach down model, then it may support `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)` and `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)`.

82 Figure 4 shows a sample flow for `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)` and `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)` messages.

83 **Figure 4 — Sample flow for VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST) and VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)**

84



85 Table 9 defines the `TCP_ReqPayload` field of Table 5 for `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)`. This specification defines a set of services under the standards body of DMTF. Other standards bodies or vendors may also define services. The `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)` is designed for the Requester to announce its supported DMTF-defined services to the Responder. Other standards bodies or vendors may define their own `VENDOR_DEFINED_REQUEST` for the Requester to announce their services to the Responder.

86 **Table 9 — VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST) TCP_ReqPayload field**

Byte offset	Field	Size (bytes)	Description
0	AvailableDmtfServices	32	Shall be a bit mask of 256 bits. Each bit represents whether the Requester can provide a DMTF-defined service identified by the `ServiceID` in Table 13 (1 means the Requester supports the service identified by this `ServiceID`; 0 means this `ServiceID` is reserved or the Requester does not support the service identified by this `ServiceID`). The upper byte of the `ServiceID` for a valid DMTF-defined service is always 0x00. A `ServiceID` is represented by the bit offset of the `AvailableDmtfServices` bit mask. For example, Bit [0] of Byte [0] of `AvailableDmtfServices` represents `ServiceID` 0x00 (ANY_REQUEST). If a Requester supports DMTF-defined services, then the Requester should support ANY_REQUEST, and hence this bit of the `AvailableDmtfServices` bit mask should be 1. Other examples: Bit [2] of Byte [0] of `AvailableDmtfServices` represents `ServiceID` 0x02 (PROVISION_CERTIFICATE); Bit [6] of Byte [0] represents `ServiceID` 0x06 (PROVISION_REFERENCE_MEASUREMENTS); and so on.
32	Reserved	3	Reserved.
35	State	1	Shall indicate the current state of the requested service. - 0x00: This is the first `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)`, or the last requested service of the Responder has been completed and the Responder may respond with a new service request. - 0x01: The `SessionMethod` field (see Table 11) of the `SessionOption` in the last `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)` response is invalid. - 0x02: The `SlotID` field of the `SessionOption` in the last `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)` response is invalid. - 0x03: The `ServiceID` (see Table 12) in the last `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)` response is invalid, because the requested service is not available on the Requester. - 0x04: The `ServiceID` in the last `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)` response is invalid, because the requested service requires a secure session but the Responder has no provisioned certificates, public keys, or pre-shared keys for establishing a session. - 0x05: The `ServiceParam` in the last `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)` response is invalid. - other values: reserved. If `State` is not 0x00, the Responder should resend the `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)` response with correct parameters.

87 Table 10 defines the `TCP_RespPayload` field of Table 7 for `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)`. The `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)` response is designed to accommodate and convey requested services defined by not only DMTF, but also other standards bodies or vendors.

88 **Table 10 — VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST) TCP_RespPayload field**

Byte offset	Field	Size (bytes)	Description
0	SessionOption	1	Shall be the Responder's preference for session for this service. See Table 11.

Byte offset	Field	Size (bytes)	Description
1	Reserved	1	Reserved.
2	ServiceRequest	variable	Shall indicate the service being requested. The service request shall follow the format defined in Table 12.

89 **Table 11 — Session option for VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)**

Bit offset	Field	Description
[2:0]	SessionMethod	Shall be the Responder's preference for the session and its establishment method. - 000b: The Responder does not provide a preference for whether a session should be established or not. - 100b: The Responder suggests that a session should be established, but does not provide a preference for using `KEY_EXCHANGE` or `PSK_EXCHANGE`. - 110b: The Responder suggests establishing a session using `PSK_EXCHANGE`. - 111b: The Responder suggests establishing a session using `KEY_EXCHANGE`. - other values: reserved.
3	Reserved	Reserved.
[7:4]	SlotID	Meaningful only if `SessionMethod` is 111b. Shall be the Responder's certificate Slot ID that the Responder suggests the Requester to use in `KEY_EXCHANGE`. Use 1111b if Responder prefers to use the Responder's public key previously provisioned to the Requester for `KEY_EXCHANGE`.

90 **Table 12 — Format of a service request**

Byte offset	Field	Size (bytes)	Description
0	SVH	svh_len	Shall follow the format of the "standards body or vendor-defined header (SVH)" defined by DSP0274. The `ServiceID` below is within the name space defined by this standards body or vendor. For DMTF, Table 13 defines a set of services under `ID` = DMTF (0x00) and `VendorIDLen` = 0x00 (no vendor ID).
svh_len	ServiceID	2	Shall specify the service being requested. For DMTF, Table 13 defines a set of services. This field shall be 0xFFFF if the Responder has no service requests.
svh_len + 2	ServiceParam	1	Shall specify the parameter for the service identified by `ServiceID`. For DMTF-defined services, the parameters are specified in Table 13 with the corresponding service.
svh_len + 3	Reserved	5	Reserved. Standards bodies or vendors may introduce additional parameters for their services.

91 The `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)` response message contains details of the Responder's request for service (for example, certificate provisioning) and preferences (for example, slot ID for the certificate to be provisioned). The Responder may request for only one service in one response. After a service is completed, the

Requester should send `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)` again to inquire about additional service requests.

92 If the Responder's `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)` response contains invalid data (such as a `ServiceID` not supported by the Requester), the Requester may resend the `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)` with error information in the `State` field. Note that this may happen after other SPDM message exchanges. For example, the Responder's `ServiceRequest` contains `ServiceID` of 0x02 (VERIFY_MEASUREMENTS; see Table 13), which requires a mutually-authenticated session. However, from the `DIGESTS` response, Requester realizes that the Responder has no provisioned certificates, public keys, or pre-shared keys for establishing a session. In this case, the Requester should send resend `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)` with `State` = 0x04.

93 Upon receiving `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)` with an error code in `State`, the Responder should correct the error and respond with a valid `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)` message. The Requester may terminate the TCP connection after a certain number (implementation-specific) of invalid `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)` responses from the Responder.

94 The Responder may request for the same service multiple times with different `ServiceParam` values (see Table 13) in multiple `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)` and `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)` pairs. For example, the Responder may request the Requester to provision multiple certificates with different values of the `Param1` field.

95 Some services require a secure session and potentially mutual authentication. Once a session is established due to a service request, the Requester may either terminate the session (by `END_SESSION`) upon completion of the service request, or keep the session active for future services and terminate the session upon completion of all services.

96 After a service request has been fulfilled, the Requester may close the TCP connection or send `VENDOR_DEFINED_REQUEST(GET_SERVICE_REQUEST)` to inquire about the Responder's additional service requests. If the Responder has no additional service requests, then the Responder may respond with `ServiceID` = 0xFFFF (see Table 13), and the Requester may close the TCP connection.

97 Table 13 specifies DMTF-defined services that the Responder may request from the Requester.

98 **Table 13 — DMTF-defined services and parameters**

`ServiceID` and short name	Description (the Responder asks the Requester to…)	`ServiceParam`
0x0000 ANY_REQUEST	Perform a service that is not specified in `VENDOR_DEFINED_RESPONSE(SERVICE_REQUEST)`. The Responder shall use this `ServiceID` if the service to be performed was negotiated with the Requester using an out-of-band method. The Responder may use this `ServiceID` to notify the Requester that the Responder is ready for SPDM communication. The Requester may send `GET_SUPPORTED_EVENT_TYPES` and `SUBSCRIBE_EVENT_TYPES` and wait for events from the Responder. The Requester should support ANY_REQUEST.	Reserved

`ServiceID` and short name	Description (the Responder asks the Requester to…)	`ServiceParam`
0x0001 SESSION_ONLY	Set up a session using the configurations specified in `SessionOption`. The Requester and the Responder may exchange (non-SPDM) application data or proprietary data within the session.	Reserved
0x0002 PROVISION_CERTIFICATE	Provision a certificate with `GET_CSR` and `SET_CERTIFICATE`.	Shall be the Responder's suggested value for `Param1` of `SET_CERTIFICATE`.
0x0003 VERIFY_MEASUREMENTS	Get Responder's measurements with `GET_MEASUREMENTS`, verify measurements, and provide results with `VENDOR_DEFINED_RESPONSE(VERIFICATION_RESULTS)`. This service shall be conducted within a session with mutual authentication, because the Responder shall be able to verify that the verification results were originated from a trustworthy Requester.	Reserved
0x0004 GET_MEASUREMENTS	Get Responder's measurements with `GET_MEASUREMENTS` without providing verification results. Use case example: the Responder reports its measurements to the Requester periodically as part of a recurring verification, and the Requester enforces policies (such as blocking network access) depending on verification results.	- Bits [7:4]: Reserved. - Bits [3:0]: A value of 0000b, 0001b, 0010b, 0011b, 0100b, 0101b, 0110b, 0111b, or 1111b shall indicate the Responder's suggested value for Bits [3:0] (Responder's certificate slot ID or provisioned public key) of `SlotIDParam` of `GET_MEASUREMENTS`. A value of 1110b shall indicate that `SlotIDParam` is negotiated using an out-of-band method or shall be determined by the Requester. Other values are reserved.
0x0005 GET_MEL	Get the Responder's measurement event log (MEL) with `GET_MEASUREMENTS` where `DMTFSpecMeasurementValueType[6:0]` == 0x08 (hash-extended measurement) and `GET_MEASUREMENT_EXTENSION_LOG`. Use of the MEL is specific to implementation and out of scope of this specification.	Reserved
0x0006 PROVISION_REFERENCE_MEASUREMENTS	Provision reference measurements with `VENDOR_DEFINED_RESPONSE(SET_REFERENCE)`.	Reserved
0x0007 PROVISION_VERIFICATION_POLICY	Provision measurement verification policy with `VENDOR_DEFINED_RESPONSE(SET_POLICY)`.	Reserved
0xFFFF INVALID	Not to provide additional services. This `ServiceID` does not represent any service.	Reserved
other	Reserved	Reserved

7.2 VENDOR_DEFINED_REQUEST(VERIFICATION_RESULTS) and VENDOR_DEFINED_RESPONSE(VERIFICATION_RESULTS_ACK)

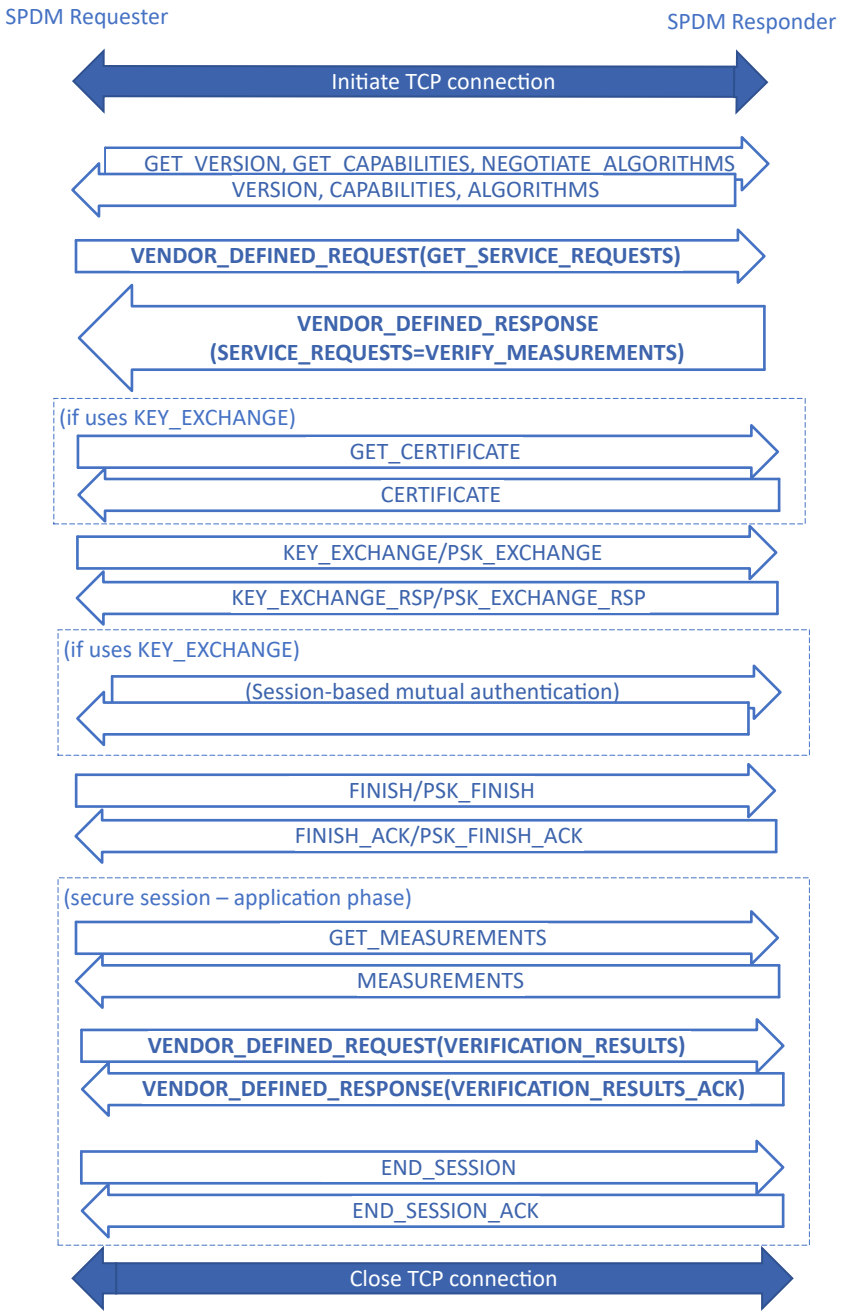
100 To fulfill the service request “VERIFY_MEASUREMENTS”, the Requester compares the measurements reported by the Responder in `MEASUREMENTS` response against reference measurements according to verification policies. The Requester should acquire reference measurements and verification policies from trusted sources. In particular endpoint that implements SPDM Requester may also implement SPDM Responder and acquire Reference Measurements and Verification Policies using mechanisms defined in this specification. The Requester shall convey the results of the verification to the Responder by sending `VENDOR_DEFINED_REQUEST(VERIFICATION_RESULTS)`. The Responder can then take proper actions based on the results.

101 The Requester shall support `VENDOR_DEFINED_REQUEST(VERIFICATION_RESULTS)` and `VENDOR_DEFINED_RESPONSE(VERIFICATION_RESULTS_ACK)` if it supports the “VERIFY_MEASUREMENTS” service. The Responder shall support `VENDOR_DEFINED_REQUEST(VERIFICATION_RESULTS)` and `VENDOR_DEFINED_RESPONSE(VERIFICATION_RESULTS_ACK)` if it uses the “VERIFY_MEASUREMENTS” service.

102 Figure 5 demonstrates a sample flow for the Requester to acquire measurements from the Responder using `GET_MEASUREMENTS` and then provide verification results to the Responder using `VENDOR_DEFINED_REQUEST(VERIFICATION_RESULTS)`. Note that this service shall be conducted within a mutually-authenticated session (that is established by either `KEY_EXCHANGE` with mutual authentication or `PSK_EXCHANGE`), because the Responder shall be able to verify that the results were originated from a trustworthy Requester.

103 **Figure 5 — Flow for Requester verifying Responder-reported measurements**

104



105 Table 14 defines the TCP_ReqPayload field of Table 5 for VENDOR_DEFINED_REQUEST(VERIFICATION_RESULTS).

106 **Table 14 — VENDOR_DEFINED_REQUEST(VERIFICATION_RESULTS) TCP_ReqPayload field**

Byte offset	Field	Size (bytes)	Description
0	Results	128	Shall be an array of up to 256 verification results, each 4 bits, corresponding to the up to 256 indexes of the `MEASUREMENTS` response, respectively. For example, Bits [3:0] of Byte [0] is for index 0; Bits [7:4] of Byte [0] is for index 1, and so on. A verification result of a measurement is represented by 4 bits, as follows: - 0000b: Measurement was not provided or not suitable for verification (such as index 0x00 (reserved)). - 0001b: A reference value for this measurement is not available to the Requester. - 0010b: A policy for this measurement is not available to the Requester. - 0011b: A reference value and a policy for this measurement are not available to the Requester. - 0101b: Requester is unable to provide verification due to other reasons. - 1000b: Measurement has passed verification. - 0100b: Measurement has failed verification. - other values: reserved. If the Responder receives a reserved value for a measurement, the Responder shall treat the verification result as “failed”.

107 The `VENDOR_DEFINED_RESPONSE(VERIFICATION_RESULTS_ACK)` message has no payload (`TCP_RespPayloadLen` = 0x0000 and the `TCP_RespPayload` field shall be absent in Table 7).

108 7.3 VENDOR_DEFINED_REQUEST(SET_REFERENCE) and VENDOR_DEFINED_RESPONSE(SET_REFERENCE_ACK)

109 The Requester sends a `VENDOR_DEFINED_REQUEST(SET_REFERENCE)` message to the Responder for the service “PROVISION_REFERENCE_MEASUREMENTS” in Table 13. If the data structure of the reference measurements guarantees the integrity, authenticity, and freshness of the reference, then the message may be sent outside of a session. Otherwise, the message shall be sent within a mutually-authenticated session.

110 The Requester shall support `VENDOR_DEFINED_REQUEST(SET_REFERENCE)` and `VENDOR_DEFINED_RESPONSE(SET_REFERENCE_ACK)` if it supports the “PROVISION_REFERENCE_MEASUREMENTS” service. The Responder shall support `VENDOR_DEFINED_REQUEST(SET_REFERENCE)` and `VENDOR_DEFINED_RESPONSE(SET_REFERENCE_ACK)` if it uses the “PROVISION_REFERENCE_MEASUREMENTS” service.

111 Table 15 defines the `TCP_ReqPayload` field of Table 5 for `VENDOR_DEFINED_REQUEST(SET_REFERENCE)`.

112 **Table 15 — VENDOR_DEFINED_REQUEST(SET_REFERENCE) TCP_ReqPayload field**

Byte offset	Field	Size (bytes)	Description
0	ReferenceLen	2	Shall be the number of bytes of the `Reference` field.
2	Reserved	2	Reserved.

Byte offset	Field	Size (bytes)	Description
4	Reference	`ReferenceLen`	Shall be the reference measurements to be used by the Responder for evaluating measurements. This field shall follow the format defined in table “General opaque data format” of DSP0274, which may carry multiple versions of the reference measurements in multiple OpaqueElementData fields. The data model of reference measurements is out of scope of this specification.

113 The `VENDOR_DEFINED_RESPONSE(SET_REFERENCE_ACK)` message has no payload (`TCP_RespPayloadLen` = 0x0000 and the `TCP_RespPayload` field shall be absent in Table 7).

114 7.4 VENDOR_DEFINED_REQUEST(SET_POLICY) and VENDOR_DEFINED_RESPONSE(SET_POLICY_ACK)

115 The Requester sends a `VENDOR_DEFINED_REQUEST(SET_POLICY)` message to the Responder for the service “PROVISION_VERIFICATION_POLICY” in Table 13. If the data structure of the measurement verification policy guarantees the integrity, authenticity, and freshness of the policy, then the message may be sent outside of a session. Otherwise, the message shall be sent within a mutually-authenticated session.

116 The Requester shall support `VENDOR_DEFINED_REQUEST(SET_POLICY)` and `VENDOR_DEFINED_RESPONSE(SET_POLICY_ACK)` if it supports the “PROVISION_VERIFICATION_POLICY” service. The Responder shall support `VENDOR_DEFINED_REQUEST(SET_POLICY)` and `VENDOR_DEFINED_RESPONSE(SET_POLICY_ACK)` if it uses the “PROVISION_VERIFICATION_POLICY” service.

117 Table 16 defines the `TCP_ReqPayload` field of Table 5 for `VENDOR_DEFINED_REQUEST(SET_POLICY)`.

118 **Table 16 — VENDOR_DEFINED_REQUEST(SET_POLICY) TCP_ReqPayload field**

Byte offset	Field	Size (bytes)	Description
0	PolicyLen	2	Shall be the number of bytes of the `Policy` field.
2	Reserved	2	Reserved.
4	Policy	`PolicyLen`	Shall be the measurement verification policy to be used by the Responder for evaluating measurements against the reference. This field shall follow the format defined in table “General opaque data format” of DSP0274, which may carry multiple versions of the policy in multiple `OpaqueElementData` fields. The data model of the policy is out of scope of this specification.

119 The `VENDOR_DEFINED_RESPONSE(SET_POLICY_ACK)` message has no payload (`TCP_RespPayloadLen` = 0x0000 and the `TCP_RespPayload` field shall be absent in Table 7).

8 ANNEX A (informative) change log

8.1 Version 1.0.0 (2024-01-17)

- WIP release

122 # 9 Bibliography

123 DMTF DSP4014, *DMTF Process for Working Bodies 2.6*, https://www.dmtf.org/sites/default/files/standards/documents/DSP4014_2.6.pdf