***Intel OpenBMC SPDM Unit test Document***

Revision: 0.1

**Document Change History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Author** | **Description** |
| 05/03/2022 | 0.1 |  | First Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Contents

[1. Purpose 4](#_Toc102491799)

[2. Introduction 4](#_Toc102491800)

[3. Unit Test 4](#_Toc102491801)

[Test environment 4](#_Toc102491802)

[Test Procedure 4](#_Toc102491803)

[Dbus system command 4](#_Toc102491804)

[test tool 6](#_Toc102491805)

[4. SPDM-simulator packet analyzes 7](#_Toc102491806)

[GET\_VERSION 7](#_Toc102491807)

[GET\_CAPABILITIES 7](#_Toc102491808)

[NEGOTIATE\_ALGORITHMS 8](#_Toc102491809)

[GET\_DIGESTS 9](#_Toc102491810)

[GET\_CERTIFICATE 9](#_Toc102491811)

[CHALLENGE 10](#_Toc102491812)

[GET\_MEASUREMENTS 11](#_Toc102491813)

# Purpose

The purpose of the document is to test items/jobs of SPDM module in Intel OpenBMC. The function outside the BMC FW is not in the scope of this document.

# 2. Introduction

The Security Protocol and Data Model (SPDM) defines messages, data objects, and sequences for performing message exchanges between devices over a variety of transport and physical media. The description of message exchanges includes authentication of hardware identities, measurement for firmware identities and session

Key exchange protocols to enable confidentiality and integrity protected data communication.

# Unit Test

## Test environment

Used spdm-simulator with Aardwork.

## Test Procedure

### Dbus system command

1. Make sure SPDM device is detected.

The following command will display all mctp devices:

busctl tree xyz.openbmc\_project.MCTP\_SMBus\_PCIe\_slot

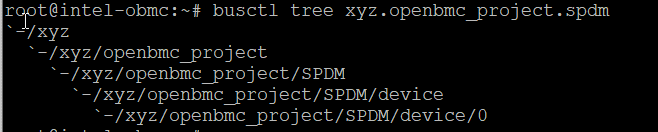
excepted result:



1. Check SPDM Daemon

The following command will display all entries of SPDM daemon:

busctl tree xyz.openbmc\_project.spdm

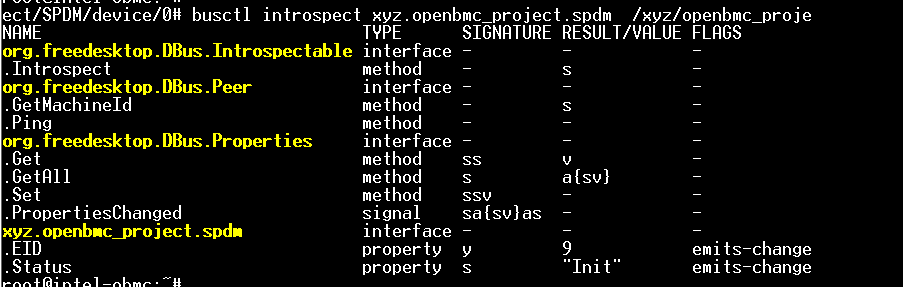


1. Check status of SPDM device before spdm-simulator start to do authenticate.

The following command will display status of SPDM device:

busctl introspect xyz.openbmc\_project.spdm /xyz/openbmc\_project/SPDM/device/0

expected result:

String of status is” Init”.

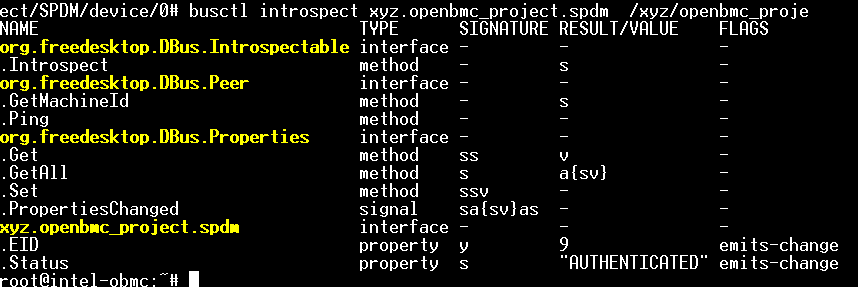
1. spdm-simulator run to do authenticate.
2. Check status of SPDM device

The following command will display status of SPDM device:

busctl introspect xyz.openbmc\_project.spdm /xyz/openbmc\_project/SPDM/device/0

expected result:

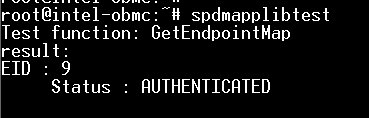
String of status is” AUTHENTICATED”.



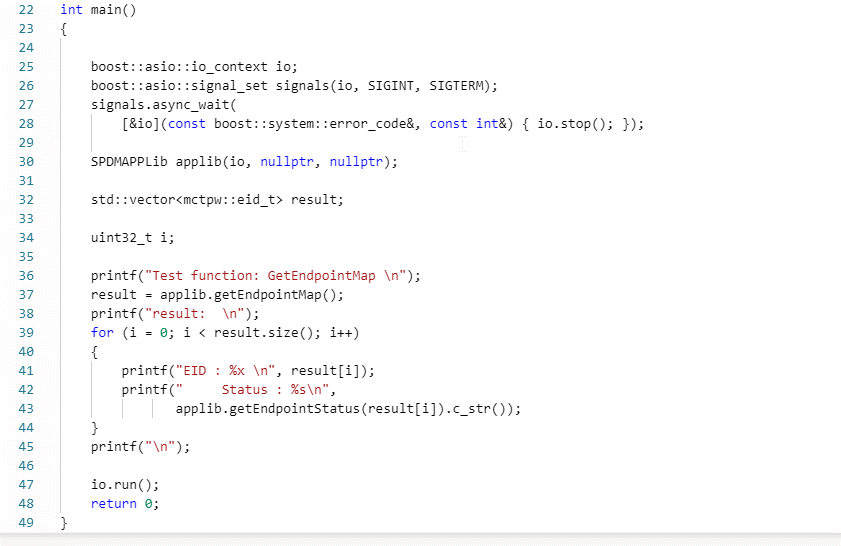
### test tool

Run test tool: spdmapplibtest

expected result:



It ‘s content is:



# SPDM-simulator packet analyzes

## GET\_VERSION

In spdm-simulator:

Request packet:



**Byte offset 1**: 0x84 = GET\_VERSION (SPDM request codes.)

Response Packet:



**Byte offset 1**: 0x04 = VERSION (SPDM response codes.)

**Byte offset 2~4**: Reserved.

**Byte offset 5: 0x03 (**Number of version entries present in this table)

**Byte offset 6~11: 0x00 0x10 0x00 0x11 0x00 0x12 (**16-bit version entry**)**

## GET\_CAPABILITIES

In spdm-simulator:

Request packet:



**Byte offset 1**: 0xE1 = GET\_CAPABILITIES (SPDM request codes.)

**Byte offset 2~4**: Reserved.

**Byte offset 5: 0x00 (**CTExponent**)**

**Byte offset 6~7**: Reserved.

**Byte offset 8~11: 0x06 0x00 0x00 0x00 (**Flags**)**

**Byte offset 12~15: 0x00 0x12 0x00 0x00 (**DataTransferSize**)**

**Byte offset 16~19: 0x00 0x12 0x00 0x00 (**MaxSPDMmsgSize**)**

Response Packet:



**Byte offset 1**: 0x61 = CAPABILITIES (SPDM response codes.)

**Byte offset 2~4**: Reserved.

**Byte offset 5: 0x00 (**CTExponent)

**Byte offset 6~7**: Reserved.

**Byte offset 8~11: 0xf7 0xfb 0x00 0x00 (**Flags**)**

**Byte offset 12~15: 0x00 0x12 0x00 0x00 (**DataTransferSize**)**

**Byte offset 16~19: 0x00 0x12 0x00 0x00 (**MaxSPDMmsgSize**)**

## NEGOTIATE\_ALGORITHMS

In spdm-simulator:

Request packet:



**Byte offset 1**: 0xE3 = NEGOTIATE\_ALGORITHMS (SPDM request codes.)

**Byte offset 2: 0x04 (**Number of algorithms structure tables in this request using ReqAlgStruct)

**Byte offset 3**: Reserved.

**Byte offset 4~5: 0x30 0x00 (**Length**)**

**Byte offset 6: 0x01 (**MeasurementSpecification)

**Byte offset 7: 0x02 (**OtherParamsSupport**)**

**Byte offset 8~11: 0x80 0x00 0x00 0x00 (**BaseAsymAlgo**)**

**Byte offset 12~15: 0x02 0x00 0x00 0x00 (**BaseHashAlgo **)**

**Byte offset 16-27**: Reserved.

**Byte offset 28: 0x00 (**ExtAsymCount)

**Byte offset 29: 0x00 (**ExtHashCount)

**Byte offset 30~31**: Reserved.

**Byte offset** **32~47:** 0x02 0x20 0x1b 0x00 0x03 0x20 0x06 0x00 0x04 0x20 0x0f 0x00 0x05 0x20 0x01 0x00 (AlgStructSize)

Response Packet:



**Byte offset 1**: 0x63 = ALGORITHMS (SPDM response codes.)

**Byte offset 2: 0x04 (**Number of algorithms structure tables in this request using RespAlgStruct)

**Byte offset 3**: Reserved.

**Byte offset 4~5: 0x34 0x00 (**Length of the response message**)**.

**Byte offset 6: 0x01 (**MeasurementSpecificationSel)

**Byte offset 7: 0x02 (**OtherParamsSelection)

**Byte offset 8~11: 0x08 0x00 0x00 0x00 (**MeasurementHashAlgo)

**Byte offset 12~15: 0x80 0x00 0x00 0x00 (**BaseAsymAlgo**)**

**Byte offset 16~19: 0x02 0x00 0x00 0x00 (**BaseHashAlgo **)**

**Byte offset 20~31**: Reserved.

**Byte offset 32: 0x00 (**ExtAsymCount)

**Byte offset 33: 0x00 (**ExtHashCount)

**Byte offset 34~35**: Reserved.

**Byte offset** **36~51:** 0x02 0x20 0x1b 0x00 0x03 0x20 0x06 0x00 0x04 0x20 0x0f 0x00 0x05 0x20 0x01 0x00 (AlgStructSize)

## GET\_DIGESTS

In spdm-simulator:

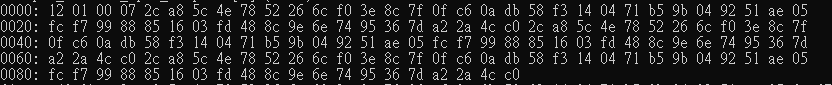
Request packet:



**Byte offset 1**: 0x81 = GET\_DIGESTS (SPDM request codes.)

**Byte offset 2~3:** Reserved.

Response Packet:



**Byte offset 1**: 0x01 = DIGESTS (SPDM response codes.)

**Byte offset 2**: Reserved.

**Byte offset 3: 0x07 (**Slot mask**)**.

**Byte offset 4~51: 0x2c 0xa8 0x5c 0x4e 0x78 0x52 0x26 0x6c 0xf0 0x3e 0x8c 0x7f 0x0f 0xc6 0x0a 0xdb 0x58 0xf3**

**0x14 0x04 0x71 0xb5 0x9b 0x04 0x92 0x51 0xae 0x05 0xfc 0xf7 0x99 0x88 0x85 0x16 0x03 0xfd 0x48 0x8c 0x9e**

**0x6e 0x74 0x95 0x36 0x7d 0xa2 0x2a 0x4c 0xc0 (**Digest[0])

**Byte offset 52~99: 0x2c 0xa8 0x5c 0x4e 0x78 0x52 0x26 0x6c 0xf0 0x3e 0x8c 0x7f 0x0f 0xc6 0x0a 0xdb 0x58**

**0xf3 0x14 0x04 0x71 0xb5 0x9b 0x04 0x92 0x51 0xae 0x05 0xfc 0xf7 0x99 0x88 0x85 0x16 0x03 0xfd 0x48 0x8c**

**0x9e 0x6e 0x74 0x95 0x36 0x7d 0xa2 0x2a 0x4c 0xc0 (**Digest[1])

**Byte offset 100~147: 0x2c 0xa8 0x5c 0x4e 0x78 0x52 0x26 0x6c 0xf0 0x3e 0x8c 0x7f 0x0f 0xc6 0x0a 0xdb 0x58**

**0xf3 0x14 0x04 0x71 0xb5 0x9b 0x04 0x92 0x51 0xae 0x05 0xfc 0xf7 0x99 0x88 0x85 0x16 0x03 0xfd 0x48 0x8c**

**0x9e 0x6e 0x74 0x95 0x36 0x7d 0xa2 0x2a 0x4c 0xc0 (**Digest[2])

## GET\_CERTIFICATE

In spdm-simulator:

Request packet:



**Byte offset 1**: 0x82 = GET\_CERTIFICATE (SPDM request codes.)

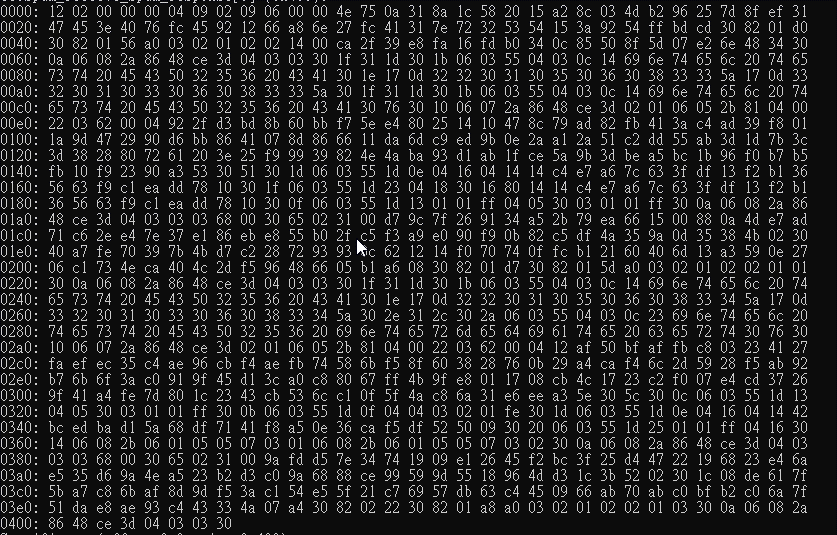
**Byte offset 2: 0x00 (**Bit [3:0]. SlotID**)**.

**Byte offset 3:** Reserved.

**Byte offset 4~5: 0x00 0x00(**Offset**)**.

**Byte offset 6~7: 0x00 0x04(**Length**)**.

Response Packet:



**Byte offset 1**: 0x02 = CERTIFICATE (SPDM response codes.)

**Byte offset 2: 0x00 (**Bit [3:0]. SlotID**)**.

**Byte offset 3:** Reserved.

**Byte offset 4~5: 0x00 0x04 (**PortionLength**)**

**Byte offset 6~7: 0x09 0x02 (**RemainderLength**)**

**Byte offset 8~1031:**

0x09 0x06 0x00 0x00 0x4e …… 0x08 0x2a 0x86 0x48 0xce 0x3d 0x04 0x03 0x03 0x30 **(**CertChain)

## CHALLENGE

In spdm-simulator:

Request packet:



**Byte offset 1**: 0x83 = CHALLENGE (SPDM request codes.)

**Byte offset 2: 0x00 (**Bit [3:0]. SlotID**)**.

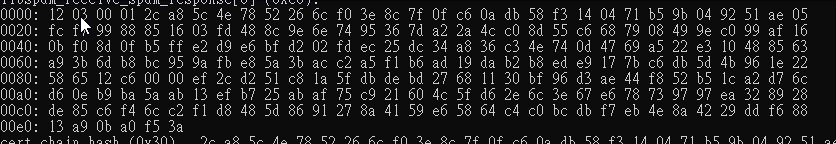
**Byte offset 3: 0xff (**Type of measurement summary hash requested. 0xFF. All measurements.).

**Byte offset 4~35: 0x0a 0xe3 0xec 0x5d 0xf4 0x43 0xf2 0xd0 0x9f 0x45 0xba 0x17 0x2c 0xe1 0x1a 0x3c 0xc8 0x67**

**0xb5 0xbd 0xd3 0x7d 0x72 0x55 0x85 0xf2 0x99 0x9f 0x5f 0x9d 0x56 0xca**

**(**The Requester should choose a random value.**)**.

Response Packet:



**Byte offset 1**: 0x03 = CHALLENGE\_AUTH (SPDM response codes.)

**Byte offset 2: 0x00 (**Response Attribute Field**)**.

**Byte offset 3: 0x01 (**Slot mask).

**Byte offset 4~53: 0x2c 0xa8 0x5c 0x4e 0x78 0x52 0x26 0x6c 0xf0 0x3e 0x8c 0x7f 0x0f 0xc6 0x0a 0xdb 0x58 0xf3**

**0x14 0x04 0x71 0xb5 0x9b 0x04 0x92 0x51 0xae 0x05 0xfc 0xf7 0x99 0x88 0x85 0x16 0x03 0xfd 0x48 0x8c 0x9e**

**0x6e 0x74 0x95 0x36 0x7d 0xa2 0x2a 0x4c 0xc0 (**CertChainHash**)**

**Byte offset 54~85: 0x8d 0x55 0xc6 0x68 0x79 0x08 0x49 0x9e 0xc0 0x99 0xaf 0x16 0x0b 0xf0 0x8d 0x0f 0xb5**

**0xff 0xe2 0xd9 0xe6 0xbf 0xd2 0x02 0xfd 0xec 0x25 0xdc 0x34 0xa8 0x36 0xc3**

**(**Responder-selected random value.**)**

**Byte offset 86~133:**

0x4e 0x74 0x0d 0x47 0x69 0xa5 0x22 0xe3 0x10 0x48 0x85 0x63 0xa9 0x3b 0x6d 0xb8 0xbc 0x95 0x9a 0xfb 0xe8

0x5a 0x3b 0xac 0xc2 0xa5 0xf1 0xb6 0xad 0x19 0xda 0xb2 0xb8 0xed 0xe9 0x17 0x7b 0xc6 0xdb 0x5d 0x4b 0x96

0x1e 0x22 0x58 0x65 0x12 0xc6 **(**MeasurementSummaryHashrtChain)

**Byte offset 134~135: 0x00 0x00(**OpaqueDataLength**)**

**Byte offset 136~231: 0xef 0x2c 0xd2 0x51 0xc8 0x1a 0x5f 0xdb 0xde 0xbd 0x27 0x68 0x11 0x30 0xbf 0x96 0xd3**

**0xae 0x44 0xf8 0x52 0xb5 0x1c 0xa2 0xd7 0x6c 0xd6 0x0e 0xb9 0xba 0x5a 0xab 0x13 0xef 0xb7 0x25 0xab 0xaf**

**0x75 0xc9 0x21 0x60 0x4c 0x5f 0xd6 0x2e 0x6c 0x3e 0x67 0xe6 0x78 0x73 0x97 0x97 0xea 0x32 0x89 0x28 0xde**

**0x85 0xc6 0xf4 0x6c 0xc2 0xf1 0xd8 0x48 0x5d 0x86 0x91 0x27 0x8a 0x41 0x59 0xe6 0x58 0x64 0xc4 0xc0 0xbc**

**0xdb 0xf7 0xeb 0x4e 0x8a 0x42 0x29 0xdd 0xf6 0x88 0x13 0xa9 0x0b 0xa0 0xf5 0x3a**

**(**Signature**)**

## GET\_MEASUREMENTS

In spdm-simulator:

Request packet:



**Byte offset 1**: 0xE0 = GET\_MEASUREMENTS (SPDM request codes.)

**Byte offset 2: 0x00 (**Request attributes.**)**.

**Byte offset 3: 0x00 (**Measurement operation. A value of 0x0 shall query the Responder for the total number of

measurement blocks available.**)**.

Response Packet:



**Byte offset 1**: 0x60 = MEASUREMENTS (SPDM response codes.)

**Byte offset 2: 0x07 (**total number of measurement indices on the device**)**.

**Byte offset 3: 0x00 (**Bit [5:4]. content changed. 00b : the Responder does not support detection of runtime

measurement changes, or this message does not contain a signature. ).

**Byte offset 4: 0x00 (**NumberOfBlocks**)**

**Byte offset 5~7: 0x00 0x00 0x00 (**MeasurementRecordLength.**)**

**Byte offset 8~39:**

0xdb 0x57 0x2d 0xb4 0x79 0xdd 0xce 0xea 0x51 0x8d 0xa2 0xbe 0x33 0x79 0x22 0x72 0x79 0xc5 0x6e 0x00 0x4e

0x2b 0x9e 0xc8 0xa6 0x8b 0x91 0x3a 0x8f 0xfa 0x28 0x5b **(**Nonce)

**Byte offset 40~41: 0x00 0x00(**OpaqueDataLength**)**