Unified Extensible Firmware Interface  
Engineering Change Request (ECR)

*Draft for Review*

Title: SPDM Protocol

Document: UEFI Specification

Sponsor: Intel

*Submission for Review Date:*

Review Approval Date: *x/xx/200x*

Submission for Technical Editing Date: *x/xx/200x*

Submission for Draft Review Date: *x/xx/200x*

Verification Date: *x/xx/200x*

Verifier: *firstname lastname, company*

Summary

## Summary of Change

**[Background]**

Today, there is new requirement to not only verify an executable image, but a device on the system. The entity to verify the device might be a standalone platform Root-of-Trust, or the system firmware once the system firmware becomes part of Chain-of-Trust. The Distributed Management Task Force (DMTF) defines **Secure Protocol and Data Model (SPDM)** specification. The hardware device standard group (such as PCI-SIG, USB, etc) defines the interface to transport the SPDM message for device authentication and measurement. Trusted Computing Group (TCG) also defines the event log for SPDM measurement.

As such, the system firmware needs an architecture way to verify a device on the platform by the device driver via the SPDM message.

**[Proposal]**

This ECR adds a new protocol - **EFI\_SPDM\_PROTOCOL**. It is similar to the Storage Security Command Protocol, or any command passthru protocol, such as SCSI PassThru Protocol, ATA PassThru Protocol, NVMe PassThru Protocol, SD MMC PassThru Protocol. This protocol abstracts the SPDM message transportation between the system firmware and the SPDM device such as PCI device, USB device, and more in the future.

The device driver (such as PCI Bus, USB Bus, etc) need produce the SPDM protocol if the device has SPDM message capability. The system firmware device security driver may consume the SPDM protocol to get the device identity, authenticate the device based upon the platform policy, get the device measurement and extend to the TPM Platform Configuration Register.

**[Tech Background – SPDM message]**

The SPDM specification defined the Requester and Responder. The Requester could be system firmware. The Responder could be the device to be authenticated or measured.

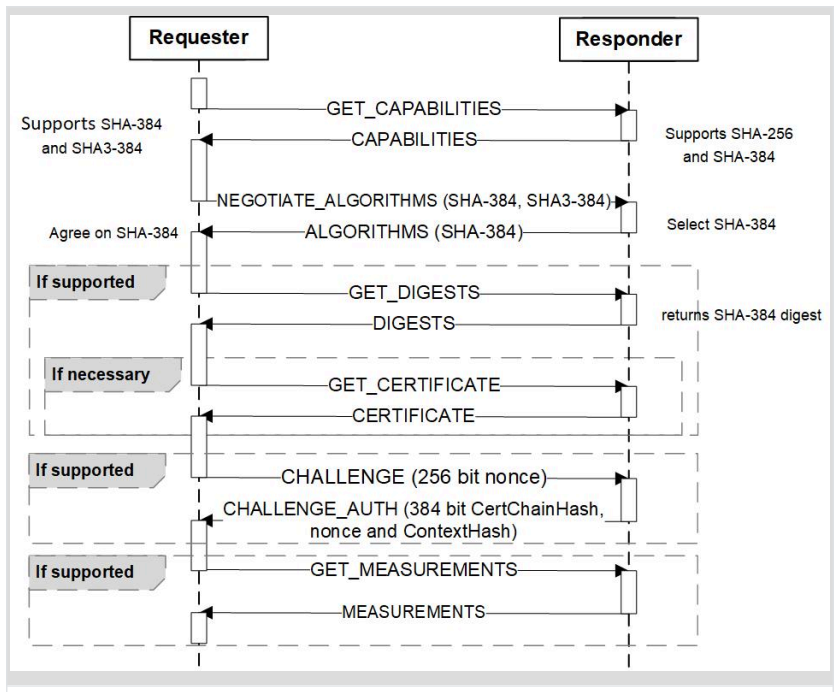
****

Figure 1 – *SPDM Message (Source: SPDM Specification)*

The SPDM specification defines a set of messages for the communication between the system firmware and the device. See table 1. The SPDM message can be on top of any device specific protocol, such as PCI, USB, SMBUS, etc. The detail of the device specific way to transport the SPDM message is defined in the device specific standard, such as PCI standard and USB standard.

Table 1 – *SPDM Message (Source: SPDM Specification)*

|  |  |  |
| --- | --- | --- |
| **Command/Response** | **Category** | **Description** |
| GET\_VERSION/VERSION | Capability Discovery and Negotiation | retrieve an endpoint's SPDM version |
| GET\_CAPABILITIES/CAPABILITIES | retrieve an endpoint's security capabilities |
| NEGOTIATE\_ALGORITHMS/ALGORITHMS | negotiate cryptographic algorithms |
| GET\_DIGESTS/DIGEST | Hardware Identity Authentication | retrieve the certificate chain digests |
| GET\_CERTIFICATE/ CERTIFICATE | retrieve the certificate chains |
| CHALLENGE/ CHALLENGE\_AUTH | authenticate an endpoint through the challenge-response protocol |
| GET\_MEASUREMENTS/MEASUREMENTS | Firmware  Measurement | retrieve firmware measurements |

**[Tech Background – UEFI Firmware (EDKII) Design]**

In EDKII, the device driver produces the SPDM\_PROTOCOL during initialization. Then device driver consumes the DEVICE\_SECURITY\_PROTOCOL to verify the device. A SpdmDeviceSecurityDxe, which produces DEVICE\_SECURITY\_PROTOCOL, consumes the SPDM\_PROTOCOL to do the device authentication or measurement. As such, we can have a common SpdmDeviceSecurityDxe driver to verify all devices, as long as the device follows the SPDM specification. See figure 2 and figure 3.

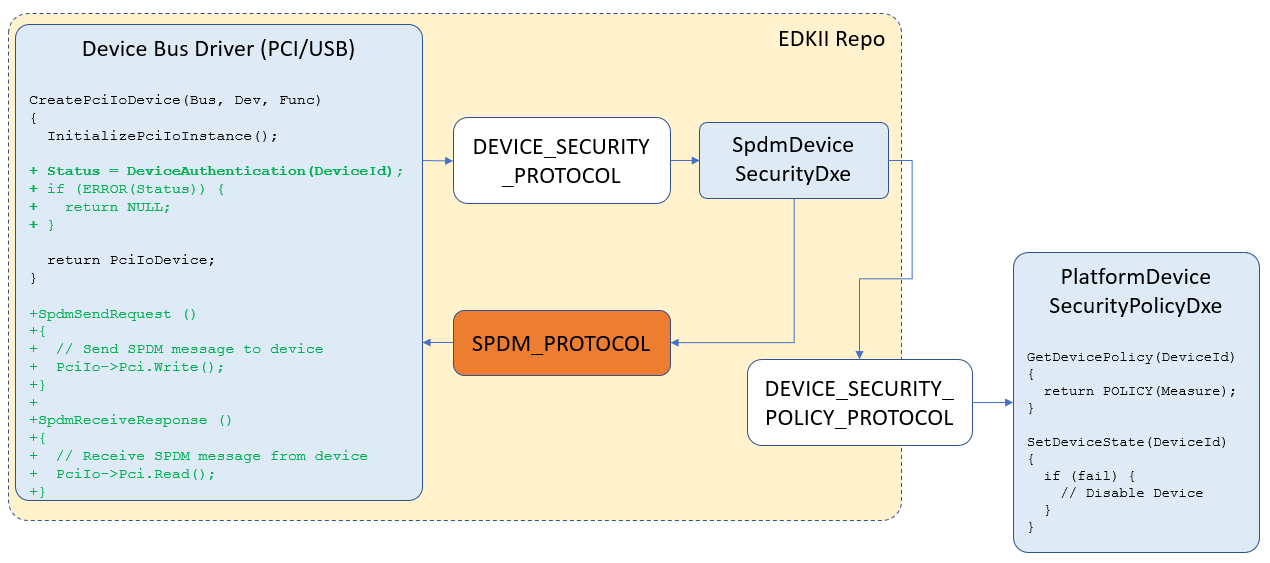


Figure 2 – *SPDM protocol usage in the EDKII (Device Authentication Component)*

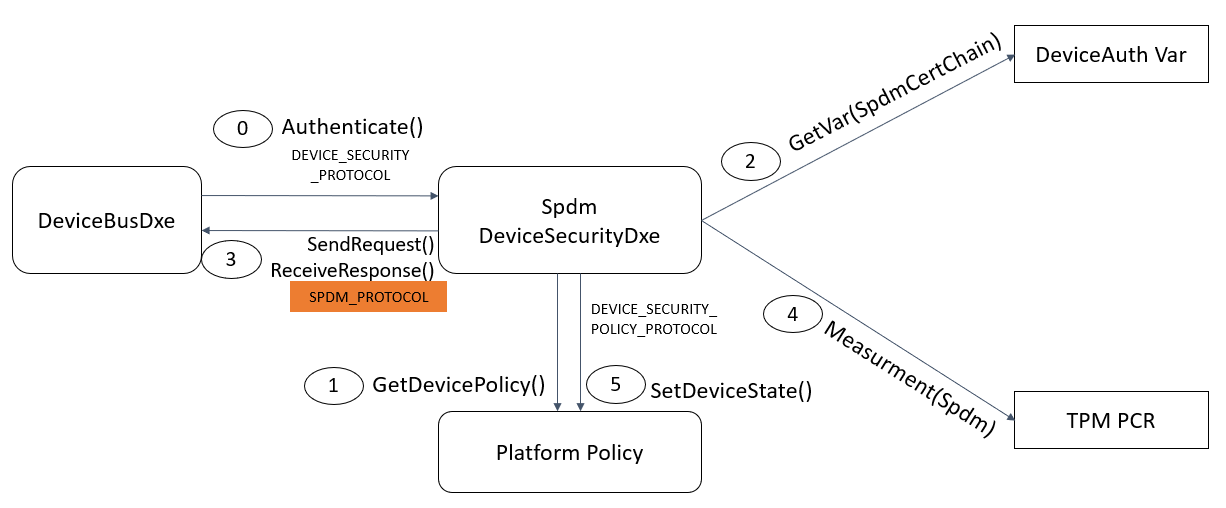


Figure 3 – *SPDM protocol usage in the EDKII (Device Authentication Flow)*

## Impact of Change

Add a new protocol in UEFI specification Chapter 32 – Secure Boot and Driver Signing.

No impact to the existing protocol.

## References

1. UEFI Specification 2.8 - www.uefi.org
2. DMTF “Security Protocol and Data Model Specification” - https://www.dmtf.org/standards/pmci
3. USB “USB Authentication Specification” - https://www.usb.org/documents
4. PCI-SIG “Component Measurement and Authentication (CMA) ECR” - https://pcisig.com/specifications/review-zone
5. TCG “TCG PC Client Platform firmware Profile Specification” - https://trustedcomputinggroup.org/resource/pc-client-specific-platform-firmware-profile-specification/, https://trustedcomputinggroup.org/wp-content/uploads/TCG\_PCClient\_PFP\_r1p05\_05\_3feb20.pdf
6. NIST SP800-193 “Platform Firmware Resiliency Guidelines” - https://csrc.nist.gov/publications/sp800

Detailed Description of the Change  
and Special Instructions

**2.6.2 Platform-Specific Elements**

**……**

35. If a platform includes an SPDM capable device, then the EFI\_SPDM\_PROTOCOL must be implemented.

**2.6.3 Driver-Specific Elements**

**……**

28. If a driver is written for an SPDM capable device, then the EFI\_SPDM\_PROTOCOL must be implemented.

32.7 SPDM Protocol

This section defines the Secure Protocol and Data Model (SPDM) protocol. This protocol is used to abstract the SPDM capable device defined in the DMTF SPDM specification to allow code running in the EFI boot services environment to send SPDM request message and receive SPDM response message to the SPDM device without specific knowledge of the type of device or controller that manages the device, such as a PCI device or a USB device. Functions are defined to send or receive SPDM message defined data to and from the SPDM capable devices.

Because the SPDM message is used to authenticate the device, the SPDM protocol producer and consumer must be aware that the SPDM message might be sent to a malicious device. Care must be taken when the driver parses the output from the SPDM capable device.

**EFI\_SPDM\_PROTOCOL**

**Summary**

This protocol provides ability to send or receive SPDM message to or from the SPDM capable device.

**GUID**

**#define EFI\_SPDM\_PROTOCOL\_GUID \**

**{0x78cad8aa, 0x7f4, 0x4c9b, {0x95, 0x5e, 0x85, 0x73, 0x2e, 0x37, 0x21, 0x96}**

**Protocol Interface Structure**

**typedef struct \_EFI\_SPDM\_PROTOCOL {**

**EFI\_SPDM\_SEND\_REQUEST SendRequest;**

**EFI\_SPDM\_RECEIVE\_RESPONSE ReceiveResponse;**

**} EFI\_SPDM\_PROTOCOL;**

**Parameters**

*SendRequest*

This service sends the SPDM request message to the SPDM capable device. See the **SendRequest()** function description.

*ReceiveResponse*

This service receives the SPDM response message from the SPDM capable device. See the **ReceiveResponse()** function description.

**Description**

The **EFI\_SPDM\_PROTOCOL** is used to send or receive the SPDM message to or from a SPDM capable device. **SendRequest()** function sends the SPDM request message, and **ReceiveResponse()** function receives the SPDM response message. The device driver must produce the **EFI\_SPDM\_PROTOCOL** if the device is capable of sending or receiving SPDM message. The platform security driver may consume the **EFI\_SPDM\_PROTOCOL** to identify the device, authenticate the device or get the measurement of the device. The **EFI\_SPDM\_PROTOCOL** must be installed to the same device handle, that produces the **EFI\_DEVICE\_PATH\_PROTOCOL** and the device access protocol, such as **EFI\_PCI\_IO\_PROTOCOL** or **EFI\_USB\_IO\_PROTOCOL** in the EFI boot services environment.

**EFI\_SPDM\_PROTOCOL.SendRequest()**

**Summary**

This function sends the SPDM request message to the SPDM capable device.

**Prototype**

**typedef**

**EFI\_STATUS**

**(EFIAPI \*EFI\_SPDM\_SEND\_REQUEST) (**

**IN EFI\_SPDM\_PROTOCOL \*This,**

**IN UINTN RequestSize,**

**IN VOID \*Request,**

**IN UINT64 Timeout**

**);**

**Parameters**

*This*

Indicate the calling context.

*RequestSize*

Size in bytes of the request message.

*Request*

A pointer to the buffer to store the request message.

*Timeout*

A The timeout, in 100ns units, to use for the sending of the SPDM request message. A *Timeout* value of 0 means that this function will wait indefinitely for the SPDM request message to send. If *Timeout* is greater than zero, then this function will return **EFI\_TIMEOUT** if the time required to send the SPDM request message is greater than *Timeout*.

**Description**

The **SendRequest()** function sends the SPDM *Request* message to the SPDM capable device.

If the *Request* is NULL or the *RequestSize* is zero, the function shall return **EFI\_INVALID\_PARAMETER**.

If the *Timeout* is non-zero and the *Request* fails to be sent within the *Timeout* period, the function shall return **EFI\_TIMEOUT**.

If the *Request* is sent successfully, the function shall return **EFI\_SUCCESS**.

If a device error occurs when the *Request* is sent to the device, the function shall return **EFI\_DEVICE\_ERROR**.

**Status Codes Returned**

|  |  |
| --- | --- |
| EFI\_SUCCESS | The function sends the SPDM request message successfully. |
| EFI\_INVALID\_PARAMETER | If the *Request* is NULL or the *RequestSize* is zero. |
| EFI\_TIMEOUT | If the *Timeout* is non-zero and the *Request* fails to be sent within the *Timeout* period. |
| EFI\_DEVICE\_ERROR | If a device error occurs when the *Request* is sent to the device. |

**EFI\_SPDM\_PROTOCOL.ReceiveResponse()**

**Summary**

This function receives the SPDM response message from the SPDM capable device.

**Prototype**

**typedef**

**EFI\_STATUS**

**(EFIAPI \*EFI\_SPDM\_RECEIVE\_RESPONSE) (**

**IN EFI\_SPDM\_PROTOCOL \*This,**

**IN OUT UINTN \*ResponseSize,**

**IN OUT VOID \*Response,**

**IN UINT64 Timeout**

**);**

**Parameters**

*This*

Indicate the calling context.

*ResponseSize*

On input, size in bytes of the destination buffer to store the response message.

On output, size in bytes of the received response message.

*Response*

A pointer to the destination buffer to store the response message.

*Timeout*

A The timeout, in 100ns units, to use for the receiving of the SPDM response message. A *Timeout* value of 0 means that this function will wait indefinitely for the SPDM response message to receive. If *Timeout* is greater than zero, then this function will return **EFI\_TIMEOUT** if the time required to receive the SPDM response message is greater than *Timeout*.

**Description**

The **ReceiveResponse()** function receives the SPDM *Response* message from the SPDM capable device.

If the *Response* is NULL, the *ResponseSize* is NULL or the *\*RequestSize* is zero, the function shall return **EFI\_INVALID\_PARAMETER**.

If the *Timeout* is non-zero and the *Response* fails to be received within the *Timeout* period, the function shall return **EFI\_TIMEOUT**.

On input, the *\*ResponseSize* indicates the size of the destination buffer to store the response message. On output, if the size of the *Response* message is smaller than or equal to *\*ResponseSize*, the *\*ResponseSize* is updated to the size of the received *Response* message and the *Response* is filled with the SPDM response data, the function shall return **EFI\_SUCCESS**.

If the size of the SPDM response data is bigger than*\*ResponseSize*, the *\*ResponseSize* is updated to the size of the SPDM response data, the function shall return **EFI\_BUFFER\_TOO\_SMALL**.

If a device error occurs when the *Response* is received from the device, the function shall return **EFI\_DEVICE\_ERROR**.

**Status Codes Returned**

|  |  |
| --- | --- |
| EFI\_SUCCESS | The function receives the SPDM response message successfully. The SPDM response data is stored in the *Response* and *\*ResponseSize* is the size of the stored *Response* message. |
| EFI\_BUFFER\_TOO\_SMALL | If the size of the SPDM response data is bigger than*\*ResponseSize*. The *\*ResponseSize* is updated to the size of the SPDM response data. |
| EFI\_INVALID\_PARAMETER | If the *Response* is NULL, the *ResponseSize* is NULL or the *\*ResponseSize* is zero. |
| EFI\_TIMEOUT | If the *Timeout* is non-zero and the *Response* fails to be received within the *Timeout* period. |
| EFI\_DEVICE\_ERROR | If a device error occurs when the *Response* is received from the device |

Addition/Change to original request