**# Title**: Confidential Computing Extension

**# Status**: Submitted to industry standard forum

**# Document**: A standalone document

**# License**: SPDX-License-Identifier: CC-BY-4.0

**# Submitter**: [TianoCore Community](<https://www.tianocore.org>)

**# Summary of the change**

**[Background]**

We are adding confidential computing (CC) virtual firmware interface in EDKII. Those interfaces will be consumed by OS. Current implementation is for Intel Trust Domain Extension (TDX). It may be used for other CC solution such as ARM Realm.

However, we cannot use TCG defined interface for TPM, because the CC environment might not produce TPM-like hardware capability. Instead the CC environment may define its own measurement register, such as TDX Runtime Measurement Register.

**[Proposal]**

This proposal adds

1. A new protocol - **EFI\_CC\_MEASUREMENT\_PROTOCOL**. This protocol abstracts the measurement extension and report in UEFI environment. It is similar to EFI\_TCG2\_PROTOCOL defined in [TCG EFI Protocol specification].
2. A new ACPI table - **CCEL**. This ACPI table abstracts the measurement event log report. It is similar to TPM2 ACPI table defined in [TCG ACPI specification].

Reference:

1. Intel TDX GHCI Specification - <https://www.intel.com/content/www/us/en/developer/articles/technical/intel-trust-domain-extensions.html>
2. UEFI Specification 2.9 - [www.uefi.org](http://www.uefi.org)
3. ACPI Specification 6.4 - [www.uefi.org](http://www.uefi.org)
4. TCG EFI Protocol specification - <https://trustedcomputinggroup.org/resource/tcg-efi-protocol-specification/>
5. TCG ACPI specification - <https://trustedcomputinggroup.org/resource/tcg-acpi-specification/>
6. TCG PFPI specification - <https://trustedcomputinggroup.org/resource/pc-client-specific-platform-firmware-profile-specification/>
7. EDKII - <https://github.com/tianocore/edk2>

**# Benefits of the change**

The OS loader may use the protocol to extend measurement and event log in CC environment.

The OS kernel may use the ACPI table to get measurement event log.

**# Impact of the change**

This is an optional protocol. It will be only available in virtual firmware confidential computing environment.

# Detailed description of the change [normative updates]

## Confidential Computing Measurement

In confidential computing, a virtual firmware may support measurement and event log based upon the hardware Trusted Execution Environment (TEE) capability. We need way to provide the abstraction for the hardware TEEs, such as Intel Trust Domain Extension (TDX). This document describes the interface between virtual firmware and virtual guest OS.

The interfaces defined in this document are optional. If the virtual firmware does not support hardware TEE based confidential computing, then the interfaces are not required. If the virtual firmware implements a virtual TPM, then the TCG defined event log, EFI\_TCG2\_PROTOCOL and TPM2 ACPI table should be used, while the interfaces defined in this document should not be published.

### Virtual Platform CC Event Log

If a virtual firmware with confidential computing (CC) capability supports measurement and an event is created, virtual firmware should report the event log with the same data structure in TCG Platform Firmware Profile (PFP) specification with **EFI\_TCG2\_EVENT\_LOG\_FORMAT\_TCG\_2** format.

The index created by the virtual firmware in the event log should be the index for the confidential computing (CC) measurement register. It is CC vendor specific. Table 1 shows the register mapping for Intel TDX, where MRTD means Trust Domain Measurement Register and RTMR means Runtime Measurement Register.

Table 1: CC-Event-Log-PCR-Index Interpretation for TDX

|  |  |
| --- | --- |
| **CC Measurement Register Index** | **TDX-measurement register** |
| 0 | MRTD |
| 1 | RTMR[0] |
| 2 | RTMR[1] |
| 3 | RTMR[2] |
| 4 | RTMR[3] |

### EFI\_CC\_MEASUREMENT\_PROTOCOL

If a virtual firmware with confidential computing (CC) capability supports measurement, the virtual firmware should produce EFI\_CC\_MEASUREMENT\_PROTOCOL with new GUID **EFI\_CC\_MEASUREMENT\_PROTOCOL\_GUID** to report event log and provide hash capability.

**EFI\_CC\_MEASUREMENT\_PROTOCOL**

**Summary**

This protocol abstracts the confidential computing (CC) measurement operation in UEFI guest environment.

**GUID**

**#define EFI\_CC\_MEASUREMENT\_PROTOCOL\_GUID \**

**{0x96751a3d, 0x72f4, 0x41a6, {0xa7, 0x94, 0xed, 0x5d, 0xe, 0x67, 0xae, 0x6b }}**

**Protocol Interface Structure**

**typedef struct \_EFI\_CC\_MEASUREMENT\_PROTOCOL {**

**EFI\_CC\_GET\_CAPABILITY GetCapability;**

**EFI\_CC\_GET\_EVENT\_LOG GetEventLog;**

**EFI\_CC\_HASH\_LOG\_EXTEND\_EVENT HashLogExtendEvent;**

**EFI\_CC\_MAP\_PCR\_TO\_MR\_INDEX MapPcrToMrIndex;**

**} EFI\_CC\_MEASUREMENT\_PROTOCOL;**

**Parameters**

*GetCapability*

Provide protocol capability information and state information. See the **GetCapability()** function description.

*GetEventLog*

Allow a caller to retrieve the address of a given event log and its last entry. See the **GetEventLog()** function description.

*HashLogExtendEvent*

Provide callers with an opportunity to extend and optionally log events without requiring knowledge of actual CC command. See the **HashLogExtendEvent()** function description.

*MapPcrToMrIndex*

Provide callers information on TPM PCR to CC measurement register (MR) mapping. See the **MapPcrToMrIndex()** function description.

**Description**

The **EFI\_CC\_MEASUREMENT\_PROTOCOL** is used to abstract the CC measurement related action in CC UEFI guest environment.

**EFI\_CC\_MEASUREMENT\_PROTOCOL.GetCapability**

**Summary**

This service provides protocol capability information and state information.

**Prototype**

**typedef**

**EFI\_STATUS**

**(EFIAPI \*EFI\_CC\_GET\_CAPABILITY)(**

**IN EFI\_CC\_MEASUREMENT\_PROTOCOL   \*This,**

**IN OUT EFI\_CC\_BOOT\_SERVICE\_CAPABILITY \*ProtocolCapability**

**);**

**Parameters**

*This*

The protocol interface pointer.

*ProtocolCapability*

The caller allocates memory for an EFI\_CC\_BOOT\_SERVICE\_CAPABILITY structure and sets the size field to the size of the structure allocated. The callee fills in the fields with the EFI protocol capability information and the current EFI CC state information up to the number of fields which fit within the size of the structure passed in.

**Description**

This function provides protocol capability information and state information.

**Status Code Returned**

|  |  |
| --- | --- |
| EFI\_SUCCESS | Operation completed successfully. |
| EFI\_INVALID\_PARAMETER | One or more of the parameters are incorrect.  The *ProtocolCapability* variable will not be populated. |
| EFI\_DEVICE\_ERROR | The command was unsuccessful.  The *ProtocolCapability* variable will not be populated. |
| EFI\_BUFFER\_TOO\_SMALL | The *ProtocolCapability* variable is too small to hold the full response. It will be partially populated (required Size field will be set). |

**Related Definitions**

**typedef struct {**

**//**

**// Allocated size of the structure**

**//**

**UINT8                            Size;**

**//**

**// Version of the EFI\_CC\_BOOT\_SERVICE\_CAPABILITY structure.**

**// For this version of the protocol,**

**// the Major version shall be set to 1**

**// and the Minor version shall be set to 0.**

**//**

**EFI\_CC\_VERSION                   StructureVersion;**

**//**

**// Version of the EFI TD protocol.**

**// For this version of the protocol,**

**// the Major version shall be set to 1**

**// and the Minor version shall be set to 0.**

**//**

**EFI\_CC\_VERSION                   ProtocolVersion;**

**//**

**// Supported hash algorithms**

**//**

**EFI\_CC\_EVENT\_ALGORITHM\_BITMAP    HashAlgorithmBitmap;**

**//**

**// Bitmap of supported event log formats**

**//**

**EFI\_CC\_EVENT\_LOG\_BITMAP          SupportedEventLogs;**

**//**

**// Indicate CC type**

**//**

**EFI\_CC\_TYPE                      CcType;**

**} EFI\_CC\_BOOT\_SERVICE\_CAPABILITY;**

**typedef struct {**

**UINT8 Major;**

**UINT8 Minor;**

**} EFI\_CC\_VERSION;**

**typedef UINT32     EFI\_CC\_EVENT\_LOG\_BITMAP;**

**typedef UINT32     EFI\_CC\_EVENT\_ALGORITHM\_BITMAP;**

**#define EFI\_CC\_BOOT\_HASH\_ALG\_SHA384     0x00000004**

**typedef struct {**

**UINT8 Type;**

**UINT8 SubType;**

**} EFI\_CC\_TYPE;**

**#define EFI\_CC\_TYPE\_NONE 0**

**#define EFI\_CC\_TYPE\_SEV 1**

**#define EFI\_CC\_TYPE\_TDX 2**

**EFI\_CC\_PROTOCOL.GetEventLog**

**Summary**

This service allows a caller to retrieve the address of a given event log and its last entry.

**Prototype**

**typedef**

**EFI\_STATUS**

**(EFIAPI \*EFI\_CC\_GET\_EVENT\_LOG)(**

**IN EFI\_CC\_MEASUREMENT\_PROTOCOL   \*This,**

**IN EFI\_CC\_EVENT\_LOG\_FORMAT   EventLogFormat,**

**OUT EFI\_PHYSICAL\_ADDRESS \*EventLogLocation,**

**OUT EFI\_PHYSICAL\_ADDRESS \*EventLogLastEntry,**

**OUT BOOLEAN   \*EventLogTruncated**

**);**

**Parameters**

*This*

The protocol interface pointer.

*EventLogFormat*

The type of event log for which the information is requested.

*EventLogLocation*

A pointer to the memory address of the event log.

*EventLogLastEntry*

If the event log contains more than one entry, this is a pointer to the address of the start of the last entry in the event log in memory.

*EventLogTruncated*

If the event log is missing at least one entry because one event would have exceeded the area allocated for the event, this value is set to TRUE. Otherwise, this value will be FALSE and the event log is complete.

**Description**

This function allows a caller to retrieve the address of a given event log and its last entry.

**Status Code Returned**

|  |  |
| --- | --- |
| EFI\_SUCCESS | Operation completed successfully. |
| EFI\_INVALID\_PARAMETER | One or more of the parameters are incorrect. |

**Related Definitions**

**typedef UINT32                       EFI\_CC\_EVENT\_LOG\_FORMAT;**

**#define EFI\_CC\_EVENT\_LOG\_FORMAT\_TCG\_2   0x00000002**

**EFI\_CC\_MEASUREMENT\_PROTOCOL.HashLogExtendEvent**

**Summary**

This service provides callers with an opportunity to extend and optionally log events without requiring knowledge of actual CC command.

**Prototype**

**typedef**

**EFI\_STATUS**

**(EFIAPI \*EFI\_CC\_GET\_EVENT\_LOG)(**

**IN EFI\_CC\_MEASUREMENT\_PROTOCOL   \*This,**

**IN UINT64   Flags,**

**IN EFI\_PHYSICAL\_ADDRESS DataToHash,**

**IN UINT64 DataToHashLen,**

**IN EFI\_CC\_EVENT   \*EfiTdEvent**

**);**

**Parameters**

*This*

The protocol interface pointer.

*Flags*

Bitmap providing additional information.

*DataToHash*

Physical address of the start of the data buffer to be hashed.

*DataToHashLen*

The length in bytes of the buffer referenced by *DataToHash*.

*EfiCcEvent*

Pointer to the data buffer containing information about the event.

**Description**

This function provides callers with an opportunity to extend and optionally log events without requiring knowledge of actual CC command. The extend operation will occur even if the function cannot create an event log entry.

**Status Code Returned**

|  |  |
| --- | --- |
| EFI\_SUCCESS | Operation completed successfully. |
| EFI\_INVALID\_PARAMETER | One or more of the parameters are incorrect. |
| EFI\_DEVICE\_ERROR | The command was unsuccessful. |
| EFI\_VOLUME\_FULL | The extend operation occurred, but the event could not be written to one or more event logs. |
| EFI\_UNSUPPORTED | The PE/COFF image type is not supported. |

**Related Definitions**

**//**

**// This bit is shall be set when an event shall be extended**

**// but not logged.**

**//**

**#define EFI\_CC\_FLAG\_EXTEND\_ONLY   0x0000000000000001**

**//**

**// This bit shall be set when the intent is to measure**

**// a PE/COFF image.**

**//**

**#define EFI\_CC\_FLAG\_PE\_COFF\_IMAGE  0x0000000000000010**

**typedef UINT32 EFI\_CC\_MR\_INDEX;**

**#pragma pack(1)**

**typedef struct {**

**//**

**// Size of the event header itself.**

**//**

**UINT32            HeaderSize;**

**//**

**// Header version. For this version of this specification,**

**// the value shall be 1.**

**//**

**UINT16            HeaderVersion;**

**//**

**// Index of the MR that shall be extended.**

**//**

**EFI\_CC\_MR\_INDEX   MrIndex;**

**//**

**// Type of the event that shall be extended**

**// (and optionally logged).**

**//**

**UINT32            EventType;**

**} EFI\_CC\_EVENT\_HEADER;**

**typedef struct {**

**//**

**// Total size of the event including the Size component,**

**// the header and the Event data.**

**//**

**UINT32                Size;**

**EFI\_CC\_EVENT\_HEADER   Header;**

**UINT8                 Event[1];**

**} EFI\_CC\_EVENT;**

**#pragma pack()**

**EFI\_CC\_MEASUREMENT\_PROTOCOL.MapPcrToMrIndex**

**Summary**

This service provides callers information on TPM PCR to CC measurement register (MR) mapping.

**Prototype**

**typedef**

**EFI\_STATUS**

**(EFIAPI \*EFI\_CC\_MAP\_PCR\_TO\_MR\_INDEX)(**

**IN EFI\_CC\_MEASUREMENT\_PROTOCOL   \*This,**

**IN TCG\_PCRINDEX   PcrIndex,**

**OUT EFI\_CC\_MR\_INDEX   \*MrIndex**

**);**

**Parameters**

*This*

The protocol interface pointer.

*PcrIndex*

TPM PCR index.

*MrIndex*

CC Measurement Register index.

**Description**

This function provides callers information on TPM PCR to CC measurement register (MR) mapping. This is CC vendor specific. Table 2 shows the register mapping for Intel TDX, where MRTD means Trust Domain Measurement Register and RTMR means Runtime Measurement Register.

Table 2: TPM PCR Index Mapping for TDX

|  |  |  |
| --- | --- | --- |
| **TPM PCR Index** | **CC Measurement Register Index** | **TDX-measurement register** |
| 0 | 0 | MRTD |
| 1, 7 | 1 | RTMR[0] |
| 2~6 | 2 | RTMR[1] |
| 8~15 | 3 | RTMR[2] |

**Status Code Returned**

|  |  |
| --- | --- |
| EFI\_SUCCESS | Operation completed successfully. |
| EFI\_INVALID\_PARAMETER | One or more of the parameters are incorrect. |
| EFI\_DEVICE\_ERROR | The command was unsuccessful. |
| EFI\_VOLUME\_FULL | The extend operation occurred, but the event could not be written to one or more event logs. |
| EFI\_UNSUPPORTED | The PE/COFF image type is not supported. |

**Related Definitions**

**typedef UINT32 TCG\_PCRINDEX;**

### EFI CC Final Events Table

All events generated after the invocation of **GetEventLog** SHALL be  
stored in an instance of an EFI\_CONFIGURATION\_TABLE named by the VendorGuid  
of **EFI\_CC\_FINAL\_EVENTS\_TABLE\_GUID**. The associated table contents SHALL be referenced by the VendorTableof **EFI\_CC\_FINAL\_EVENTS\_TABLE**.

**#define EFI\_CC\_FINAL\_EVENTS\_TABLE\_GUID \**

**{0xdd4a4648, 0x2de7, 0x4665, {0x96, 0x4d, 0x21, 0xd9, 0xef, 0x5f, 0xb4, 0x46}}**

**typedef struct {**

**//**

**// The version of this structure. It shall be set to 1.**

**//**

**UINT64                  Version;**

**//**

**// Number of events recorded after invocation of GetEventLog API**

**//**

**UINT64                  NumberOfEvents;**

**//**

**// List of events of type CC\_EVENT.**

**//**

**//CC\_EVENT              Event[1];**

**} EFI\_CC\_FINAL\_EVENTS\_TABLE;**

**#pragma pack(1)**

**//**

**// Crypto Agile Log Entry Format.**

**// It is similar with TCG\_PCR\_EVENT2 except MrIndex.**

**//**

**typedef struct {**

**EFI\_CC\_MR\_INDEX     MrIndex;**

**UINT32              EventType;**

**TPML\_DIGEST\_VALUES  Digests;**

**UINT32              EventSize;**

**UINT8               Event[1];**

**} CC\_EVENT;**

**#pragma pack()**

### CC Event Log ACPI Table

A virtual firmware with confidential computing (CC) capability may set up an ACPI table to pass the CC event log information. The event log created by the virtual firmware owner contains the hashes to reconstruct the confidential computing (CC) measurement registers.

Table 3: CC Event Log ACPI Table

|  |  |  |  |
| --- | --- | --- | --- |
| **Field** | **Byte Length** | **Byte Offset** | **Description** |
| Header |  |  |  |
| Signature | 4 | 0 | ‘CCEL’ Signature. |
| Length | 4 | 4 | Length, in bytes, of the entire Table |
| Revision | 1 | 8 | 1 |
| Checksum | 1 | 9 | Entire table must sum to zero. |
| OEMID | 6 | 10 | Standard ACPI header |
| OEM Table ID | 8 | 16 | Standard ACPI header |
| OEM Revision | 4 | 24 | Standard ACPI header |
| Creator ID | 4 | 28 | Standard ACPI header |
| Creator Revision | 4 | 32 | Standard ACPI header |
| CC Type | 1 | 36 | Confidential computing (CC) type.  0: Reserved  1: SEV  2: TDX |
| CC Subtype | 1 | 37 | Confidential computing (CC) type specific sub type. |
| Reserved | 2 | 38 | Reserved. Must be 0. |
| Log Area Minimum Length (LAML) | 8 | 40 | Identifies the minimum length (in bytes) of the system’s pre-boot CC event log area |
| Log Area Start Address (LASA) | 8 | 48 | Contains the 64-bit-physical address of the start of the system's pre-boot CC event log area in QWORD format. Note: The log area ranges from address LASA to LASA+(LAML-1). |

**# Special Instructions**

NO