OPEN
Compute Project

Project Cerberus Firmware Challenge Specification

Author:
Bryan Kelly, Principal Firmware Engineering Manager, Microsoft
Christopher Weimer Senior Firmware Engineer, Microsoft
Akram Hamdy Firmware Engineer, Microsoft

Revision History

Date	Description
28-08-2017	V0.01 - Initial Draft
28-09-2017	V0.02 - Add References section
28-10-2017	V0.03 - Move message exchange from protocol to register based
02-12-2018	V0.04 – Add MCTP Support and update session authentication
04-30-2018	V0.05 – Incorporate Supplier feedback
10-15-2018	V0.06 – Update Authentication flow. Change measurement to PMR and attestation integration.
01-10-2019	V0.07 – Change PMR naming to PM due to static requirements on extension.
02-15-2019	V0.08 – Add Firmware Recovery image update commands. Clarify Error Response
06-26-2019	V0.09 – Add Reset Configuration command. Identify commands subject to the cryptographic timeout.
08-05-2019	V0.10 – Update Cerberus-defined MCTP message definition.
10-21-2019	V0.11 – Add detail on Mfg pairing for devices. Add commands to get RIoT, chip, and host reset information.

Your use of this Specification may be subject to other third party rights. THIS SPECIFICATION IS PROVIDED "AS IS." The contributors expressly disclaim any warranties (express, implied, or otherwise), including implied warranties of merchantability, non-infringement, fitness for a particular purpose, or title, related to the Specification. The entire risk as to implementing or otherwise using the Specification is assumed by the Specification implementer and user. IN NO EVENT WILL ANY PARTY BE LIABLE TO ANY OTHER PARTY FOR LOST PROFITS OR ANY FORM OF INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS SPECIFICATION OR ITS GOVERNING AGREEMENT, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND WHETHER OR NOT THE OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

CONTRIBUTORS AND LICENSORS OF THIS SPECIFICATION MAY HAVE MENTIONED CERTAIN TECHNOLOGIES THAT ARE MERELY REFERENCED WITHIN THIS SPECIFICATION AND NOT LICENSED UNDER THE OWF CLA OR OWFa. THE FOLLOWING IS A LIST OF MERELY REFERENCED TECHNOLOGY: INTELLIGENT PLATFORM MANAGEMENT INTERFACE (IPMI); I^2C IS A TRADEMARK AND TECHNOLOGY OF NXP SEMICONDUCTORS ; EPYC IS A TRADEMARK AND TECHNOLOGY OF ADVANCED MICRO DEVICES INC.; ASPEED AST 2400/2500 FAMILY PROCESSORS IS A TECHNOLOGY OF ASPEED TECHNOLOGY INC.; MOLEX NANOPITCH, NANO PICOBLADE, AND MINI-FIT JR AND ASSOCIATED CONNECTORS ARE TRADEMARKS AND TECHNOLOGIES OF MOLEX LLC; WINBOND IS A TRADEMARK OF WINBOND ELECTRONICS CORPORATION; NVLINK IS A TECHNOLOGY OF NVIDIA; INTEL XEON SCALABLE PROCESSORS, INTEL QUICKASSIST TECHNOLOGY, INTEL HYPER-THREADING TECHNOLOGY, ENHANCED INTEL SPEEDSTEP TECHNOLOGY, INTEL VIRTUALIZATION TECHNOLOGY, INTEL SERVER PLATFORM SERVICES, INTEL MANAGABILITY ENGINE, AND INTEL TRUSTED EXECUTION TECHNOLOGY ARE TRADEMARKS AND TECHNOLOGIES OF INTEL CORPORATION; SITARA ARM CORTEX-A9 PROCESSOR IS A TRADEMARK AND TECHNOLOGY OF TEXAS INSTRUMENTS; GUIDE PINS FROM PENCOM; BATTERIES FROM PANASONIC. IMPLEMENTATION OF THESE TECHNOLOGIES MAY BE SUBJECT TO THEIR OWN LEGAL TERMS.

Table of Contents

List of Figures

List of Tables

Summary

Throughout this document, the term "Processor" refers to all Central Processing Unit (CPU), System On Chip (SOC), Micro Control Unit (MCU), and Microprocessor architectures. The document details the required challenge protocol required for Active Component and Platform RoTs. The Processor must implement all required features to establish a hardware based Root of Trust. Processors that intrinsically fail to meet these requirements must implement the flash protection Cerberus RoT described Physical Flash Protection Requirements document.

Active Components are add-in cards and peripherals that contain Processors, Microcontrollers or devices that run soft-logic.

This document describes the protocol used for attestation measurements of firmware for the Platform's Active RoT. The specification encompasses the pre-boot, boot and runtime challenge and verification of platform firmware integrity. The hieratical architecture extends beyond the typically UEFI measurements, to include integrity measurements of all Active Component firmware. The document describes the APIs needed to support the attestation challenge for Project Cerberus.

1 Physical Communication Channel

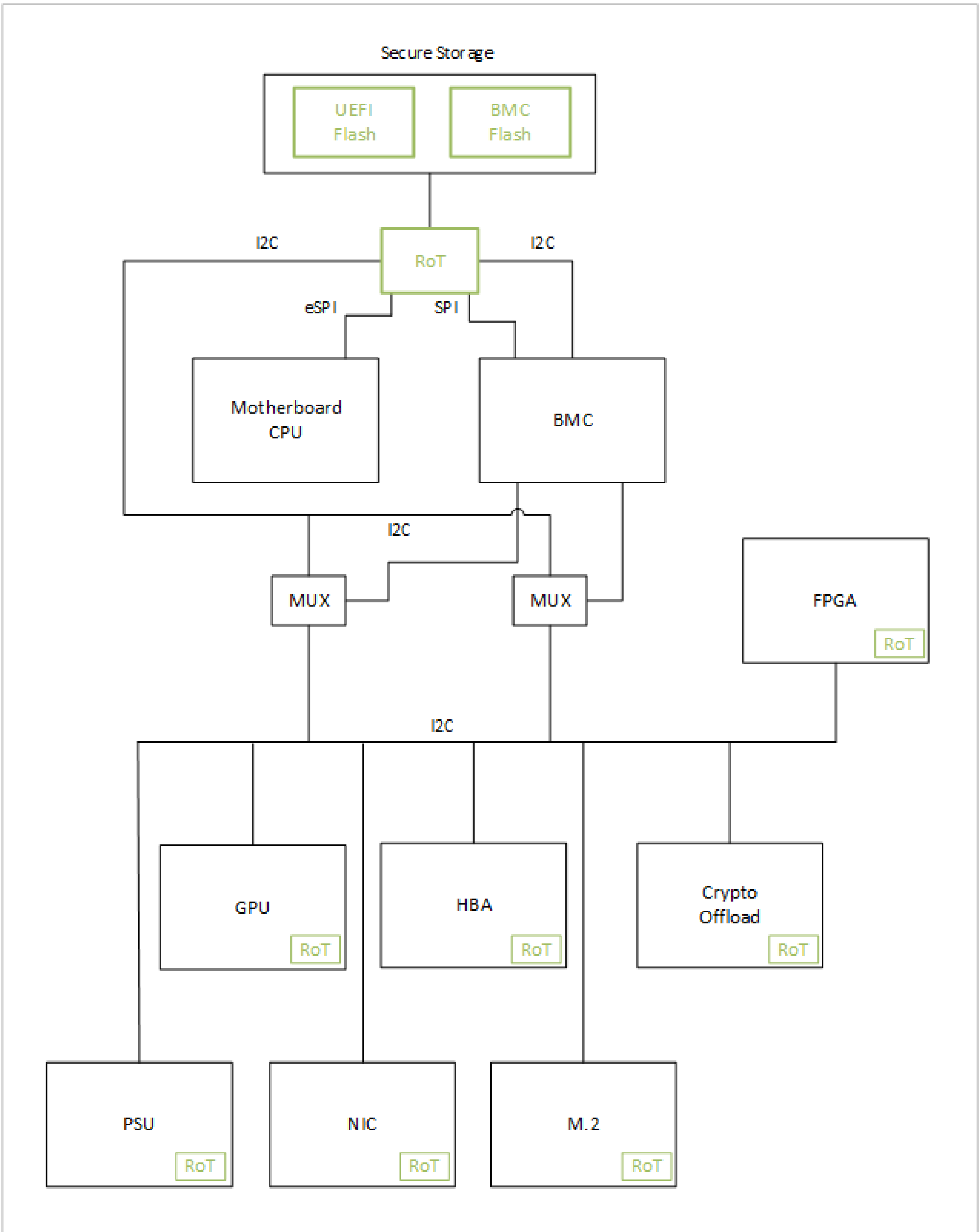
The typically cloud server motherboard layout has I2C buses routed to all Active Components. These I2C buses are typically used by the Baseboard Management Controller (BMC) for the thermal monitoring of Active Components. In the Cerberus board layout, the I2C lanes are first used by the platform Cerberus microcontroller during boot and pre-boot, then later mux switched back to the BMC for thermal management. Cerberus can at any time request for the BMC to yield control for runtime challenge and attestation. Cerberus controls the I2C mux position, and coordinates access during runtime. It is also possible for Cerberus to proxy commands through the BMC at runtime, with the option for link encryption and asymmetric key exchange, making the BMC blind to the communications. The Cerberus microcontroller on the motherboard is referred to as the Platform Active Root-of-Trust (PA-RoT). This microcontroller is head of the hierarchical root-of-trust platform design and contains an attestable hash of all platform firmware kept in the Platform Firmware Manifest (PFM) and Component Firmware Manifest.

Most cloud server motherboards route I2C to Active Components for thermal monitoring, the addition of the mux logic is the only modification to the motherboard. An alternative to adding the additional mux, is to tunnel a secure challenge channel through the BMC over I2C. Once the BMC has been loaded and attested by Cerberus, it can act as an I2C proxy. This approach is less desirable, as it limits platform attestation should the BMC ever fail attestation. In either approach, physical connectors to Active Component interfaces do not need to change as they already have I2C.

Active Components with the intrinsic security attributes described in the "Processor Secure Boot Requirements" document do not need to place the physical Cerberus microcontroller between their

Processor and Flash. Active Components that do not meet the requirements described in the "Processor Secure Boot Requirements" document are required to implement the Cerberus micro-controller between their Processor and Flash to establish the needed Root-of-Trust. Figure 1 Motherboard I2C lane diagram, represents the pre-boot and post-boot measurement challenge channels between the motherboard PA-RoT and Active Component RoTs (AC-RoT).

Figure 1 Motherboard I2C lane diagram



The Project Cerberus firmware attestation is a hierarchical architecture. Most Active Components in the modern server boot to an operational level before the platform's host processors complete their initialization and become capable of challenging the devices. In the Cerberus design, the platform is held in pre-power-on or reset state, whereby Active Components quarantined and challenged for their firmware measurements.   Active Components must respond to challenges from the PA-RoT confirming the integrity of their firmware before they are taken out of quarantine.

In this version of the Cerberus platform design, the Platform Firmware Manifest (PFM) and Component Firmware Manifests are static. The manifest is programmable through the PA-RoT's communication interface.   Auto-detection of Active Components and computation of the PFM/CFM will be considered in future version of the specification.   The PFM and CFM are manifests of allowed firmware versions and their corresponding firmware measurements.   The manifests contain a monotonic identifier used to restrict rollbacks.

The PA-RoT uses the measurements in the Component Firmware Manifest to challenge the Active Components and compare their measurements. The PA-RoT then uses the digest of these measurements as the platform level measurement, creating a hierarchical platform level digest that can attest the integrity of the platform and active component firmware.

The PA-RoT will support Authentication, Integrity and Confidentiality of messages. Active Components RoT's (AC-RoT) will support Authentication, and Integrity of messages and challenges. To facilitate this, AC-RoT are required to support certificate authentication. The Active Component will support a component unique CA signed challenge certificate for authentication.

Note: I2C is a low speed link, there is a performance tradeoff between optimizing the protocol messages and strong cryptographic hashing algorithms that carry higher bit counts.   RoT's that cannot support certificate authentication are required to support hashing algorithms and either RSA or ECDSA signatures of firmware measurements.

1.1 Power Control

In the Cerberus motherboard design, power and reset sequencing is orchestrated by the Platform's Active RoT.   When voltage is applied to motherboard, it passes through in-rush circuity to a CPLD that performs time sensitive sequencing of power rails to ensure stabilization. Once power good level is established, the platform is considered powered. Upon powering the platform in the Cerberus design, the only active component powered-on is the PA- RoT.   The RoT first securely loads and decompresses its internal firmware, then verifies the integrity of Baseboard Management Controller (BMC) firmware by measuring the BMC flash. When the BMC firmware has been authenticated the Active RoT enables power to be applied to the BMC. Once the BMC has been powered the Active RoT authenticates the firmware for the platform UEFI, during which time the RoT sequences power to the PCIe slots and begins Active Component RoT challenge.   When the UEFI has been authenticated the platform is held in system reset, the Active RoT will keep the system in reset until Active Component RoTs have responded to the measurement challenge. Any PCIe ports that do not respond to their measurement challenge will be subsequently unpowered. Should any of the expected Active Components fail to respond to the measurement challenge, Cerberus polices determine whether the system should boot with the Active Component powered off, or the platform should remain on standby power, while reporting the measurement failure to the Data Center Management Software through the OOB path.

2 Communication

The Cerberus platform Active RoT communicates with the Active Component RoT's over I2C. The protocol supports an authentication and measurement challenge. The Cerberus Platform Active RoT generates a secure asymmetric key pair unique to the microcontroller closely following the DICE architecture. Private keys are inaccessible outside of secure region of the Cerberus RoT. Key generation and chaining follows the RIoT specification, described in section: 9.3 DICE and RIoT Keys and Certificates. Derived platform alias public keys are available for use in attestation and communication from the Active Components RoT's during the challenge handshake for establishing communication.

2.1 RSA/ECDSA Key Generation

The Cerberus platform Active RoT should support the Device Identifier Composition Engine (DICE) architecture. In DICE, the Compound Device Identifier (CDI) is used as the foundation for device identity and attestation. Keys derived from the Device Unique Secret and measurement of first mutable code are used for data protection within the microcontroller and attestation of upper firmware layers. The Device Id asymmetric key pair is derived cryptographically from the CDI and associated with the Device Id Certificate. The CDI uses the Unique Device Secret derived from PUF and other random entropy including the microcontroller unique id and Firmware Security Descriptors.

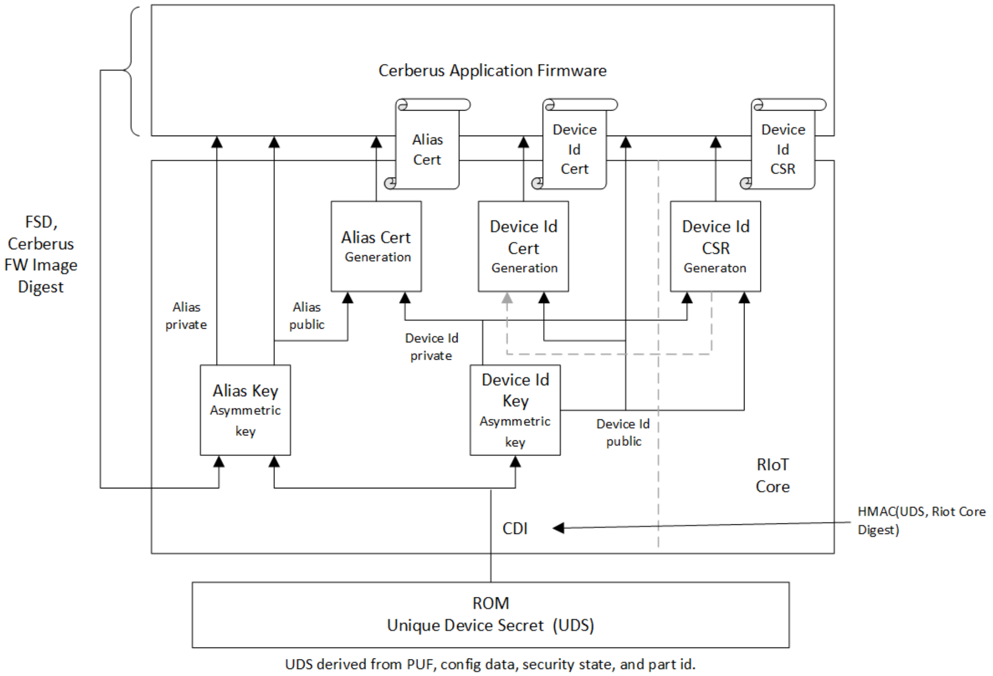
Cerberus implements RIoT Core architecture for certificate generation and attestation. For details on key generation on DICE and RIoT review section: 9.3 DICE and RIoT Keys and Certificates.

Note: The CDI is a compound key based on the Device Secret (Device Unique), Microcontroller Security Descriptors and seconds stage bootloader (mutable) measurement. The second stage bootloader is mutable code, but not typically updated with firmware updates. Changes to the second stage bootloader will result in a different CDI, resulting in different asymmetric Device Id key generation. Certificates associated with the previous Device key be invalidated, and a new Certificate would need to be signed.

A second asymmetric key pair certificate is created in the RIoT Core layer of Cerberus and passed to the Cerberus Application firmware. This key pair forms the Alias Certificate and is derived from the CDI, Cerberus Firmware Security Descriptor and measurement of the next stage Cerberus Firmware.

Proof-of-knowledge of the CDI derived private key known as the Device Id private key is used as a building-blocks in a cryptographic protocol to identify the device. The Device Id private key is used to signs the Alias Certificate, thus verifying the integrity of the key. During initial provisioning the Device Id Certificate is CA signed by Microsoft Certificate Authority. When provisioned, the Device Id keys must match the previously signed public key.

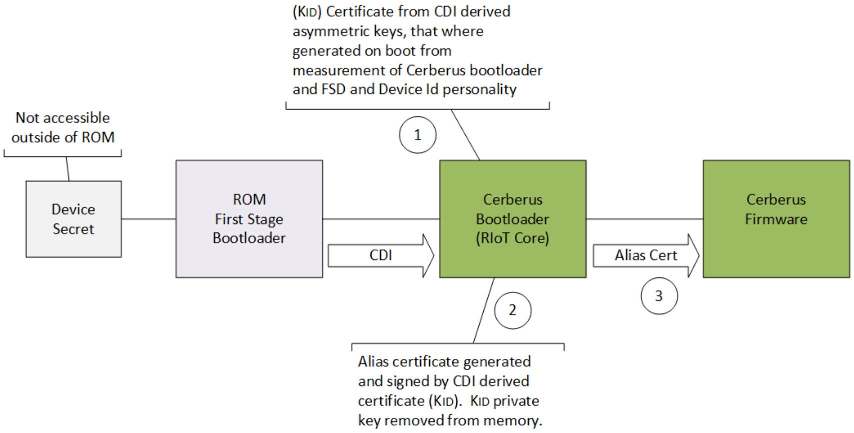
Figure 2 RioT Core Key Generation


Cerberus Application Firmware
Alias Cert
Device Id Cert
Device Id CSR
FSD, Cerberus FW Image Digest
Alias Cert Generation
Device Id Cert Generation
Device Id CSR Generaton
Alias private
Alias public
Device Id private
Alias Key Asymmetric key
Device Id Key Asymmetric key
Device Id public
RIoT Core
CDI
HMAC(UDS, Riot Core Digest)
ROM
Unique Device Secret (UDS)
UDS derived from PUF, config data, security state, and part id.

Note: The CDI and Device Id private key are security erased before exiting RIoT Core.

Each layer of the software can use its private key certificate to sign and issue a new certificate for the next layer, each successive layer continues this chain. The certificate in the application layer (Alias Certificate) can be used when authenticating with devices and establishing a secure channel. The certificate can also establish authenticity. Non-application layer private keys used to sign certificates must be accessible only to the layer they are generated. The Public keys are persisted or passed to upper layers, and eventually exchanged with upstream and downstream entities.

Figure 3 Certificate Generation

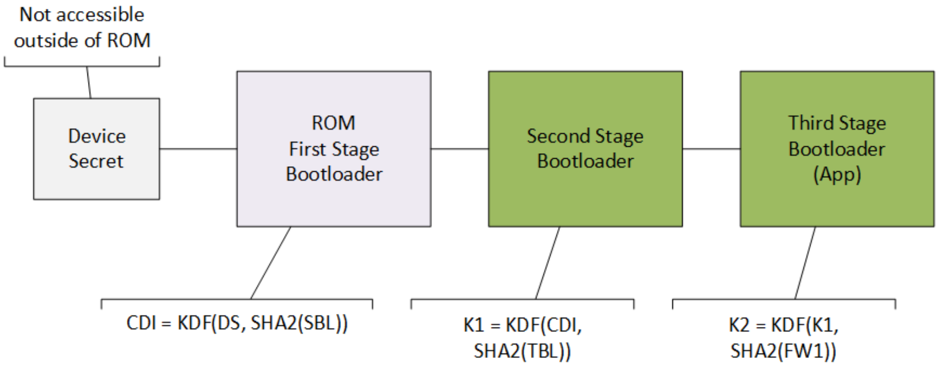

(KID) Certificate from CDI derived asymmetric keys, that where generated on boot from measurement of Cerberus bootloader and FSD and Device Id personality
Not accessible outside of ROM
1
Device Secret
ROM First Stage Bootloader
Cerberus Bootloader (RIoT Core)
Cerberus Firmware
CDI
Alias Cert
2
3
Alias certificate generated and signed by CDI derived certificate (KID). KID private key removed from memory.

2.2 Chained Measurements

The Cerberus firmware measurements based on the Device Identifier Composition Engine (DICE) architecture: https://trustedcomputinggroup.org/work-groups/dice-architectures

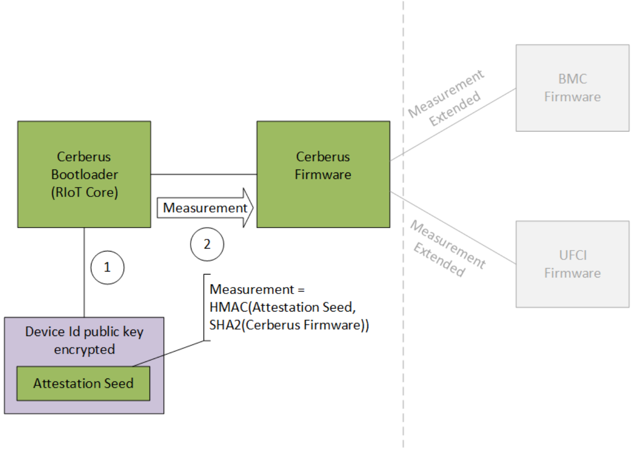
The first mutable code on the RoT is the Second Bootloader (SBL). The CDI is a measurement of the HMAC(Device Secret Key + Entropy, H(SBL)). This measurement then passes to the second stage boot loader, that calculates the digest of the Third Bootloader (TBL), on the Cerberus RoT this is the Application Firmware: HMAC(CDI, H(TBL)).

Figure 4 Measurement Calculation



The Third Stage Bootloader (TBL) which runs the Cerberus Application Firmware will take additional area measurements of the SPI/QSPI flash for the Processor it protects, measuring both active and inactive areas. The TBL measurements are verified and extended with an attestation freshness seed. The final measurement is signed, sealed, and made available to the challenge software.

Figure 5 BMC/UEFI Attestation Seed


Cerberus Bootloader (RIoT Core)
Measurement
Cerberus Firmware
1
2
Measurement = HMAC(Attestation Seed, SHA2(Cerberus Firmware))
Device Id public key encrypted
Attestation Seed
Measurement Extended
BMC Firmware
Measurement Extended
UFCI Firmware

Seeds for attesting firmware by the application firmware can be extended from Cerberus Firmware measurements, or using the Alias Certificates a dedicated freshness seed can be provided for measuring the protected processor firmware.

The measurements are stored in either firmware or hardware register values within the PA-RoT. The Seed is typically transferred to the device using either, Device Cert, Alias Cert or Attestation Cert.

3 Protocol and Hierarchy

The following section describes the capabilities and required protocol and Application Programming Interface (API) of the motherboard's Platform Active RoT (PA-RoT) and Active Component to establish a platform level RoT. The Cerberus Active RoT and Active Component RoTs are required to support the following I2C protocol.

The protocol is derived from the MCTP SMBus/I2C Transport Binding Specification. A limited version of the protocol is defined for devices that do not support MCTP. If an AC-RoT implements the Attestation Protocol over MCTP, it may also optionally implement the minimum attestation protocol over native SMBus/I2C.

3.1 Attestation Message Interface

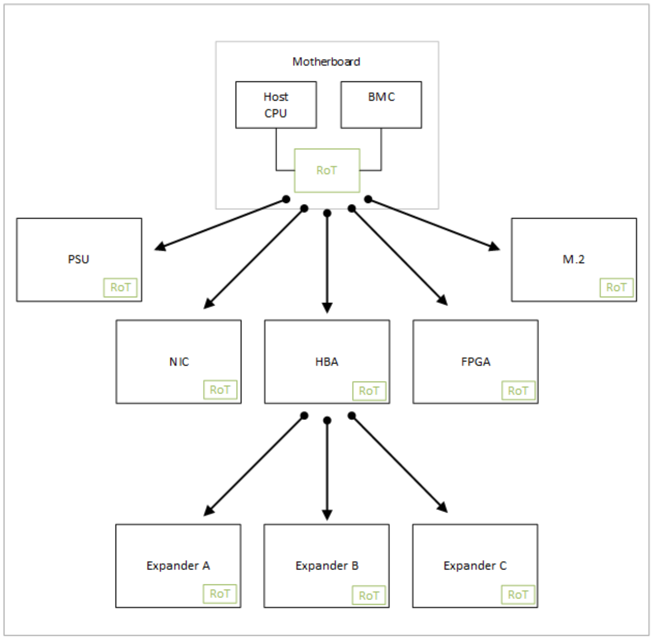
The Attestation Message Interface uses the MCTP over I2C message protocol for transporting the Attestation payloads. The AC-RoT MCTP Management Endpoint should implement the required behaviors detailed in the Management Component Transport Protocol (MCPT) Base Specification, in relation to the MCTP SMBus/I2C Transport Binding Specification. The following section outlines additional requirements upon the expected behavior of the Management Endpoint:

- The Message Interface Request and Response Messages are transported with a custom message type.
- MCTP messages will be transmitted in a synchronous Request and Response manner only. An Endpoint (AC-RoT) should never initiate a Request Message to the Controller (PA-RoT).
- MCTP Endpoints must strictly adhere to the response timeout defined in this specification. When an Endpoint receives a standard message, it should be transmitting the response within 100ms. If the Endpoint has not begun transmitting the Response Message within 100ms, it should drop the message and not respond.
- MCTP Endpoints must strictly adhere to the response timeout advertised for cryptographic commands. Cryptographic commands include transmission of messages signature generation and verification. The cryptographic command timeout multiplier is negotiated in the Device Capabilities command.
- MCTP leaves Authentication to the application implementation. This specification partially follows the flow of USB Authentication Specification flow, when authentication has been established attestation seeds can be exchanged.
- It is not required that the Management Endpoint response to ARP messages. AC-RoT Endpoints should not generate any ARP messages to Notify Master. Devices should be aware they are normally behind I2C muxes and should not master the I2C bus outside of the allotted time they are provided to response to an MCTP Request Message.
- MCTP Endpoint devices should be response only.
- Irrespective as to whether Endpoints are ARP capable, they should operate in a Non-ARP-capable manner.

- MCTP specifications use big endian byte ordering while this specification uses little endian byte ordering. This ordering does not change the payload order in which bytes are sent out on the physical layer.
- Endpoints should support Fixed Addresses; Endpoint IDs are supported to permit multiple MCTP Endpoints behind a single physical address.
- As defined in the MCTP SMBus/I2C Transport Binding Specification Endpoints should support fast-mode 400KHz.
- Endpoint devices that do not support multi-master should operate in slave mode. The PA-RoT PCD will identify the mode of the device. The Master will issue SMBUS Write Block in MCTP payload format, the master will then issue an I2C Read for the response. The Master read the initial 12 bytes and use byte 3 and the MCTP EOM header bits to determine if additional SMBUS read commands are required to collect the remainder of the response message.
- Endpoints should support EID assignment using the MCTP Set Endpoint ID control message.
- Endpoints should support the MCTP Get Vendor Defined Message Support control message to indicate support level for the Cerberus protocol.

The Platform Cerberus Active RoT is always the MCTP master. Active Component RoT's can be configured as Endpoint, or Endpoint and Master. An Active Component RoT Endpoint and Master should interface to separate physical busses. There is no requirement for master arbitration as master and slave definitions are hierarchically established. The only hierarchy whereby the Active Component RoT becomes both Endpoint and Master is when there is a downstream sub-device, such as the Host Bus Adapter (HBA) depicted in the following block diagram:

Figure 6 Root of Trust Hierarchy



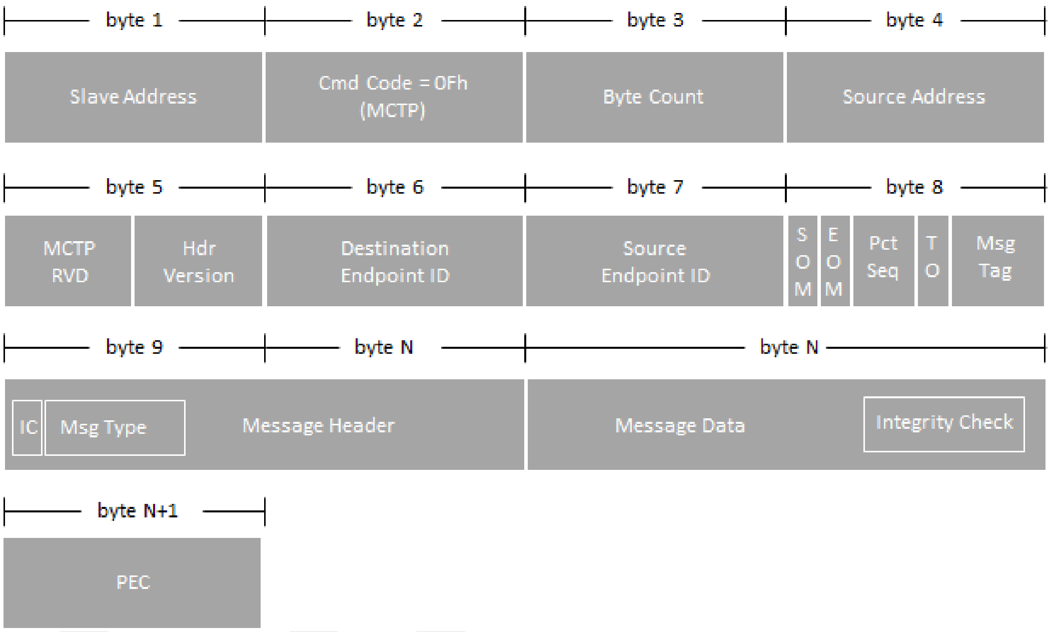
In this diagram, the HBA RoT is an Endpoint to the Platform Active RoT and Master to the downstream HBA Expanders. To the Platform's Active RoT, the HBA is an Endpoint RoT. To the HBA Expanders, the HBA Controller is a Master RoT.

The messaging protocol encompasses Management Component Transport Protocol (MCPT) Base Specification, in relation to the MCTP SMBus/I2C Transport Binding Specification, whereby the Active Component RoT is Endpoint and the Platform's Active RoT as Master.

3.2 Protocol Format

All MCTP transactions are based on the SMBus Block Write bus protocol. The following diagram shows MCTP encapsulated message.

Figure 7 MCTP Encapsulated Message



A package should contain a minimum of 1 byte of payload, with the maximum not to exceed the negotiated MCTP Transmission Unit Size.

3.3 Packet Format

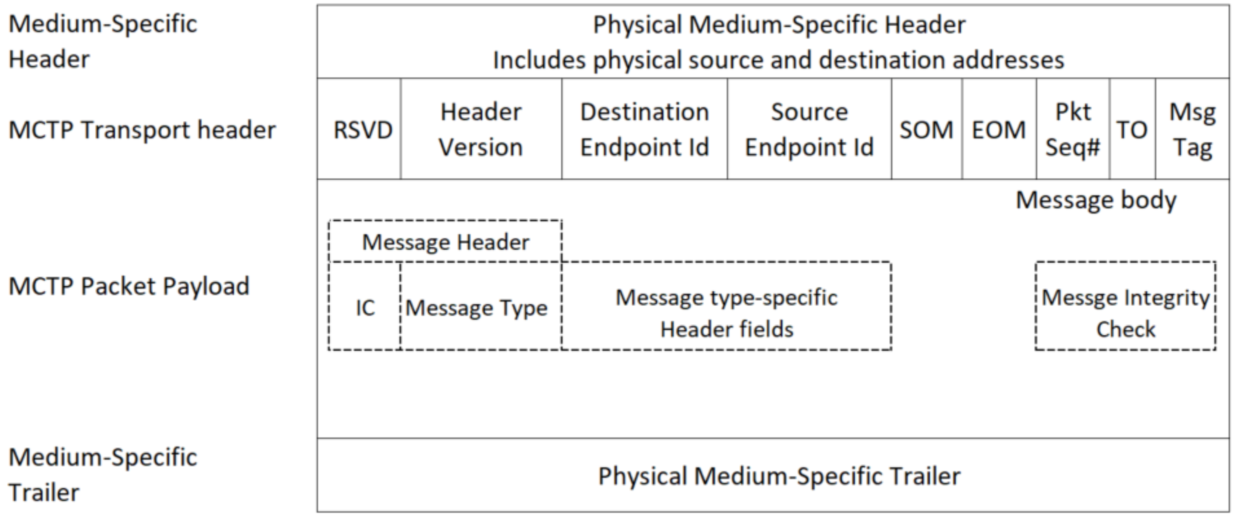
The Physical Medium-Specific Header and Physical Medium-Specific Trailer are defined by the MCTP transport binding specification utilized by the port. Refer to the MCTP transport binding specifications.

A compliant Management Endpoint shall implement all MCTP required features defined in the MCTP base specification.

The base protocol's common fields include message type field that identifies the higher layer class of message being carried within the MCTP protocol.

3.4 Transport Layer Header

Figure 8 Transport Layer Header



The Management Component Transport Protocol (MCTP) Base Specification defines the MCTP packet header (refer to DSP0236 for field descriptions). The fields of an MCTP Packet are shown in Table 1 Field Definitions.

Table 1 Field Definitions

Field Name	**Description**	**Field Size**
Medium-Specific Header	This represents the header for the protocol that encapsulates MCTP packets over a physical medium	Variable
Medium-Specific Trailer	This represents the trailer fields for the protocol that encapsulates MCTP packets over a physical medium	Variable
MCTP Transport Header	Provides version and addressing for the packet.	32 bits
RSVD	Reserved	4 bits
Header Version	Header Version Identifies the format of physical framing and data integrity.	4 bits
Destination Endpoint Id	The EID to the endpoint to receive the MCTP packet.	8 bits
Source Endpoint Id	The EID of the originator of the MCTP packet	8 bits
SOM	Start of Message is set to true (1b) for the first packet of a message.	1 bit
EOM	End of Message is set to true (1b) for the last packet of a message.	1 bit

Pkt Seq#	Packet Sequence Number for messages that span multiple packets. Increments modulo 4 on each successive packet up through the packet contained the EOM flag set.	2 bits
Message Tag	Combined with Source Endpoint Id and TO field to identify unique message at MCTP transport layer. For messages that are split up into multiple packets, the TO and Message Tag bits remain the same for all packets from the SOM to the EOM.	3 bits
TO	Tag Owner bit identifies whether the message tag was originated by the endpoint that is the source of the message or by the endpoint that is the destination of the message. MCTP message types use this for Request/Response messages.	1 bit
Message body	Payload of the MCTP message, can span multiple MCTP packets	Variable
IC	MCTP Integrity check bit 0 = No MCTP message integrity 1 = MCTP message integrity check is present	1 bit
Message Type	Defines the type of payload within the MCTP message header and data. Message type codes are defined in the MCTP ID and Codes	7 bits
Message header	Header data for the message type.	Variable
Message Data	Data for the message defined by the message type	Variable
MCTP Packet Payload	Payload of the message body carried in the packet. Limited by the transfer unit size. Review MCTP Base Specification for further details.	Variable
Message Integrity Check	Message type specific integrity check over the contest of the message body	Variable

Null (0) Source and Destination EIDs are typically supported, however AC-RoT devices that have multiple MCTP Endpoints may specify an EID value greater than 7 and less than 255. The PA-RoT does not broadcast any MCTP messages.

3.5 MCTP Messages

An MCTP message consists of one or more MCTP packets. There are typically two types of Messages, MCTP Control Messages and MCTP Command Messages. The maximum sized Command Message is 4224 bytes, while the maximum size of the Control Message is 64 bytes.

3.5.1 Message Type

The message type should be 0x7E as per the Management Component Transport Protocol (MCTP) Base Specification.   The message type is used to support Vendor Defined Messages where the Vendor is identified by the PCI based Vendor ID. The initial message header is specified in the Management Component Transport Protocol (MCTP) Base Specification, and detailed below for completeness:

Table 2 Vendor Defined Message

Message Header	Byte	
Request Data	1:2	PCI/PCIe Vendor ID. The MCTP Vendor Id formatted per 00h Vendor ID format offset.
	3:N	Vendor-Defined Message Body. 0 to N bytes.
Response Data	1:2	PCI/PCIe Vendor ID, the value is formatted per 00h Vendor ID offset
	3:M	Vendor-Defined Message Body. 0 to M bytes

The Vendor ID is a 16-bit Unsigned Integer, described in the PCI 2.3 specification. The value identifies the device manufacturer.

The message body and content is described in Table 6 MCTP Message Format.

3.5.2 Message Fields

The format of the MCTP message consists of a message header in the first two bytes, followed by the message data, and ending with the Message Integrity Check.

The Message header contains a Message Type (MT) field and Integrity Check (IC) that are defined by the MCTP Base Specification. The Message Type field indicate

3.5.3 Message Integrity Check

The Message Integrity Check field contains a 32-bit CRC computed over the contents of the message

3.5.4 Packet Assembly into Messages

An MCTP message may be split into multiple MCTP Packet Payloads and sent as a series of packets. Refer to the MCTP Base Specification for packetization and message assembly rules.

3.5.5 Request Messages

Request Messages are messages that are generated by a Master MTCP Controller and sent to an MCTP Endpoint. Request Messages specify an action to be performed by the Endpoint. Request Messages are either Control Messages or Command Messages.

3.5.6 Response Messages

Response Messages are messages that are generated when an MCTP Endpoint completes processing of a previously issued Request Message. The Response Message must be completed within the allocated time or discarded.

3.6 EID Assignment

The Platform Active RoT acts the MCTP master, and one of its roles is to assign EIDs to the different Active Component RoT devices. The EIDs to be assigned to the AC-RoTs are to be defined in the PCD manifest. All Active Component RoT devices should support the MCTP Set Endpoint ID control request and response messages. The Platform Active RoT will have a static EID of 0x0B.

4 Certificates

The PA-RoT and AC-Rot will have a minimum of two certificates: Device Id Certificate (typically CA signed by offline CA) and the Alias Certificate (signed by the Device Id Certificate). The PA-RoT may also have an additional Attestation Certificate signed by the Device Id Certificate.

Certificates follow the 9.3 DICE and RIoT Keys and Certificates with 9.4 USB Type C Authentication Specification size and Certificate chain encapsulation.

4.1 Format

All Certificates shall use the X509v3 ASN.1 structure. All Certificates shall use binary DER encoding for ASN.1. All Certificates shall use the cryptographic methods listed in

OID and Common Name attributes are defined in the 9.3 DICE and RIoT Keys and Certificates.

4.2 Textual Format

All text ASN.1 objects contained within Certificates, shall be specified as either a UTF8String, PrintableString, or IA5String. The length of any textual object shall not exceed 64 bytes excluding the DER type and DER length encoding.

4.3 Distinguished Name

The distinguished name consists of many attributes that uniquely identify the device. Distinguished name uniqueness can be accomplished by including attributes such as the serial number.

4.4 Object Identifier

Object Identifier should follow 9.3 DICE and RIoT Keys and Certificates

4.5 Serial Number

As per 9.3 DICE and RIoT Keys and Certificates, the Certificate *Serial Numbers* MUST be statistically unique per-Alias Certificate.

If the security processor has an entropy source, an 8-octet (positive) random number MAY be used.

If the security processor has does not have an entropy source, then an 8-octet (positive) *Serial Number* MAY be generated using a cryptographically secure key derivation function based on a secret key, such as those described in SP800-108 [9.6 NIST Special Publication 800-108]. The *Serial Number* MUST be unique for each generated certificate. For the Alias Certificate, this SHOULD be achieved by incorporating the FWID into the key derivation process (e.g. as the *Context* value in SP-800-108)

4.6 Certificate Chain

The Certificate Chain should conform to the lengths in the 9.4 USB Type C Authentication Specification. The maximum Certificate Chain Size is 4096 bytes.

Certificates are grouped into Certificate Chains. A Certificate Chain is the binary (byte) concatenation of the fields shown Table 3 Certificate Chain

Table 3 Certificate Chain

Offset	Field	Size	Description
0	Length	2	Total Length of the Certificate Chain in bytes, including all fields in this table.
2	Reserved	2	Set to zero
4	RootHash	32	SHA256 hash of the Root Certificate.
36	Certificates	Length - 26	One or more ANS.1 DER encoded X509v3 Certificates where first Certificate is signed by the Root and each subsequent Certificate is signed by the proceeding Certificate.

The certificate recommended cryptographic methods for interoperability are defined in Table 4 Recommended Algorithms for Interoperability

Table 4 Recommended Algorithms for Interoperability

Method	Use
X509v3, DER encoding	Certificate format
ECDSA, NIST P256, secp256r1 curve, uncompressed point	Digital signing of Certificate
SHA256	Hash algorithm

5 Authentication

A session in the context of this specification is the process of establishment of authentication, integrity and confidentiality between the external data center software and the PA-RoT. Since only a single session is supported on a given I2C channel, there are three types of session supported by the Platform RoT, secured and authenticated, secured and unsecured.

5.1 AC-RoT Authentication establishment

The initial session authentication uses certificate authentication that results measurements being exchanged.

The certificate authentication flow closely follows the USB Authentication Architecture and Authentication Messages with exception of establishing a secure connection following the certificate authentication. The Authentication Message header is not used, but the body and command flow closely follow the V1.0 USB Authentication Message Protocol. The USB Type-C Authentication Protocol version is assumed V1.0 for this specification. The USB Type-C Authentication MessageType (81h, 82h, 83h) is the Command is this specification.

Relevant sections of the specification are as follows:

Section 3 Authentication Architecture of USB Authentication Specification

Section 4 Authentication Protocol of USB Authentication Specification

Section 5 Authentication Messages of SUB Authentication Specification

The authentication sequence starts with the PA-RoT issuing a Digest Command. The AC-RoT responds with Digest command response with certificate chain digests. This allows the PA-RoT to determine if the certificate chain has been cached, and optionally skip requesting the certificate chain.

The PA-RoT then optionally issues Get Certificate Request, the AC-RoT responds with Certificate Response.

The PA-RoT then issues a Challenge command contain a RN1. The AC-RoT will respond with a Challenge Auth, containing the signed request message, authenticating that it has the private key corresponding to the signed public key exchanged earlier.

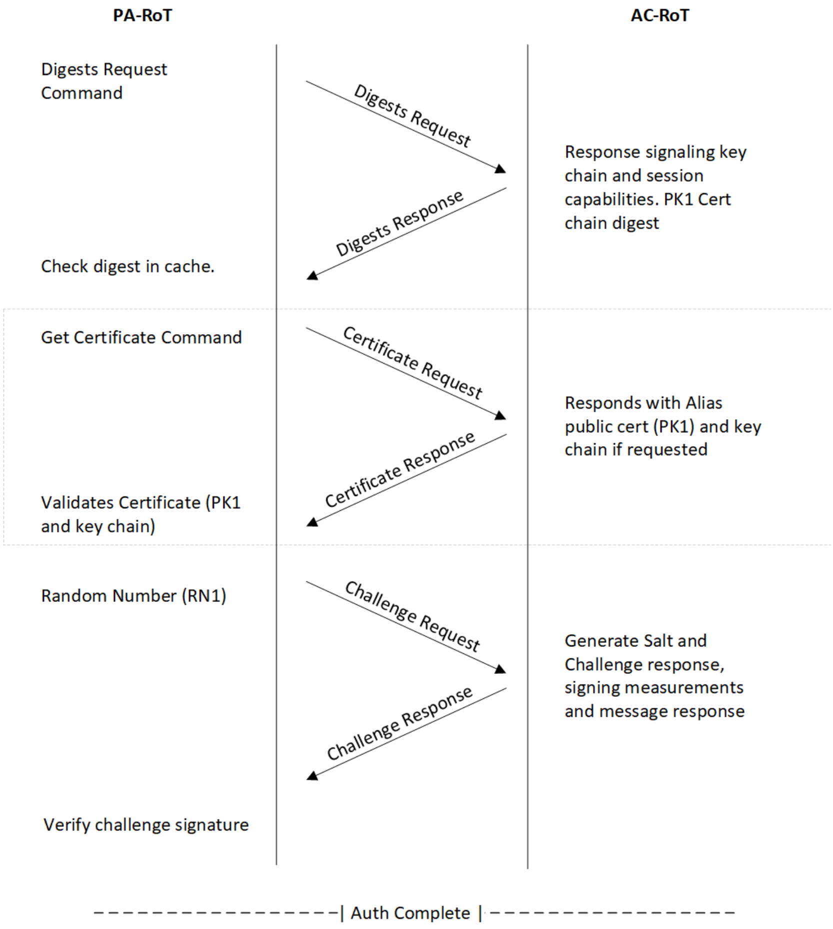
Different to the USB Authentication the challenge response is extended to include firmware measurements. This is detailed in: Table 28 CHALLENGE_AUTH Response

After querying the device capabilities, the PA-RoT will challenge the AC-RoT by issuing a Digests request. The Digests response will return a SHA256 list of the certificates in the chain. If the PA-RoT has cached the certificate chain, it may choose to not retrieve the certificate chain.

The PA-RoT may choose to issue Certificate request for Certificates in the AC-RoT chain. The AC-RoT will respond with the requested public certificate, which should have origination from a trusted CA digitally signed signature.

The PA-RoT will verify the signed certificate signature of the AC-RoT, if verification fails or certificate has been revoked, the session challenge will fail. The PA-RoT and AC-RoT can be updated with revocation patches, see firmware update specification for further details.

Figure 9 AC-RoT Authentication Flow


PA-RoT
AC-RoT
Digests Request Command
Digests Request
Response signaling key chain and session capabilities. PK1 Cert chain digest
Digests Response
Check digest in cache.
Get Certificate Command
Certificate Request
Responds with Alias public cert (PK1) and key chain if requested
Certificate Response
Validates Certificate (PK1 and key chain)
Random Number (RN1)
Challenge Request
Generate Salt and Challenge response, signing measurements and message response
Challenge Response
Verify challenge signature
Auth Complete

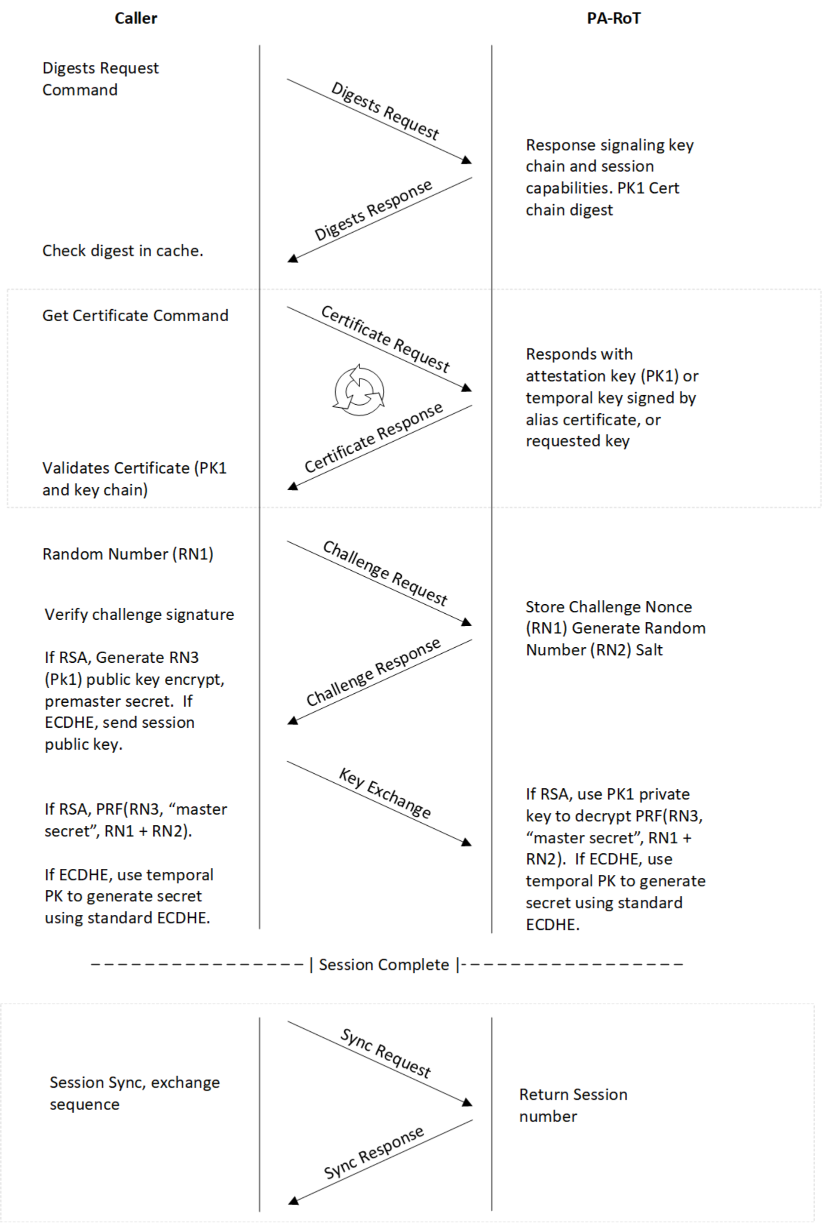
The PA-RoT uses the USB Type C Authentication flow for Certificate chain retrieval. If the PA-RoT already has the chain cached, it may skip retrieval.

It is required that the root certificate in the chain is CA signed, not self-signed.

5.2 PA-RoT Authentication establishment

The PA-RoT communication supports establishment of authentication, integrity and confidentiality between the northbound external data center software and the PA-RoT. The authentication flow is like the AC-RoT with the establishment of confidentiality. The PA-RoT authentication flow supports either RSA or ECDSA asymmetric key authentication, with either RSA or ECDHE key exchange for encrypted communication. The Device Capabilities command and certificates determine the authentication and key exchange.

Figure 10 PA-RoT Authentication


Caller
PA-RoT
Digests Request Command
Digests Request
Response signaling key chain and session capabilities. PK1 Cert chain digest
Digests Response
Check digest in cache.
Get Certificate Command
Certificate Request
Responds with attestation key (PK1) or temporal key signed by alias certificate, or requested key
Certificate Response
Validates Certificate (PK1 and key chain)
Random Number (RN1)
Challenge Request
Verify challenge signature
Store Challenge Nonce (RN1) Generate Random Number (RN2) Salt
Challenge Response
If RSA, Generate RN3 (Pk1) public key encrypt, premaster secret. If ECDHE, send session public key.
Key Exchange
If RSA, PRF(RN3, "master secret", RN1 + RN2).
If RSA, use PK1 private key to decrypt PRF(RN3, "master secret", RN1 + RN2). If ECDHE, use temporal PK to generate secret using standard ECDHE.
If ECDHE, use temporal PK to generate secret using standard ECDHE.
----------------| Session Complete |----------------
Sync Request
Session Sync, exchange sequence
Return Session number
Sync Response

After querying the PA-RoT device capabilities, the external caller can challenge the issuing a Digests request. The Digests response will return a SHA256 list of the certificates in the chain. If the caller has cached the certificate chain, it may choose to not retrieve the certificate chain.

The caller may choose to issue Certificate request for Certificates in the PA-RoT chain. The PA-RoT will respond with the requested public certificate, which should have origination from a trusted CA digitally signed signature.

The caller will verify the signed certificate signature of the PA-RoT, if verification fails or certificate has been revoked, the session challenge will fail.

The PA-RoT uses the USB Type C Authentication flow for Certificate chain retrieval. If the PA-RoT already has the chain cached, it may skip retrieval.

It is required that the root certificate in the chain is CA signed, not self-signed.

5.3 PA-RoT to AC-RoT Challenge

1. PA-RoT: Issues DIGEST command
    - AC-RoT: response with Key chain digest
2. PA-RoT: checks if key chain is cached.
    - PA-RoT skips CERTIFICATE request if key chain cached.
3. PA-RoT: issues CERTIFICATE request
    - AC-RoT responds with Certificate offset data
    - PA-RoT, if trusted root CA, verifies Cert Signature of PK1
    - PA-RoT, if not trusted root CA, continues CERTIFICATE request for chain of PK1
4. PA-RoT Issues CHALLENGE command containing nonce (^{RN1})
    - AC-RoT responds with challenge and digest signature and Salt. The Salt is used for both the message hash freshness and (^{RN2}).
    - AC-RoT provides collective firmware measurement and signature of payload.
    - PA-RoT verifies the digest and signature
    - PA-RoT stores ^{RN2}

5.4 External Software to PA-RoT Challenge

The challenge to the PA-RoT follows 5.3 PA-RoT to AC-RoT Challenge with the addition of enabling confidentiality using the following sequence:

5. Depending on the key exchange algorithm, if ECDHE temporal key is used for establishing shared secret, or the PA-RoT encrypts a nonce (^{RN3}) using PK1 from the AC-RoT. The PA-RoT then sends the encrypted nonce in Key Exchange command.
    - AC-RoT and PA-RoT both generate and encryption session key, KDF(RN3, RN2 |RN1)
6. Session Activation transmission AES encrypted based on highest level of PA-RoT device capabilities.
7. Session Sequence incrementally changes on message transactions.
8. Session sync for session state.

Note: KDF = SP800-108 Counter Mode and key is concentration of RN3, RN2 RN1.

5.5 Two Tier Authentication

After establishing the encrypted channel described in section: 5.2, the requester can send the secondary paired authentication key or request a new authentication key for unlock of key signing features and secure storage.

6 Command Format

The following section describes the MTCP message format to support the Authentication and Challenge and Attestation protocol. The Request/Response message body describes the Vendor Defined MCTP message encapsulated inside the MTCP transport. This section does not describe the MCTP Transport Header, which includes the MCTP Header Version, Destination Endpoint ID and other fields as defined by MCTP protocol. The MTCP message encapsulation is described in section: 3.2

The MCTP Get Vendor Defined Message Support command enables discovery of what Endpoint vendor defined messages are supported. The discovery identifies the vendor organization and defined messages types.

The body of the Get Vendor Defined Message Support is described in table:

Table 5 Get Vendor Defined Message Support

<table>
<tr><td colspan="8">+0</td><td colspan="8">+1</td><td colspan="8">+2</td><td colspan="8">+3</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>I
C</td><td colspan="7">Msg Type = 0</td><td colspan="3">Rsvd</td><td colspan="5">Instance ID</td><td colspan="8">Cmd Code = 06</td><td colspan="8">Comp Code = 0</td></tr>
<tr><td colspan="8">Vendor ID Set Selector = 0xFF</td><td colspan="8">Vendor ID Format = 0x00</td><td colspan="8">PCI Vendor ID[0]</td><td colspan="8">PCI Vendor ID[1]</td></tr>
<tr><td colspan="16">Command Set Type = 0x0A</td><td colspan="16"></td></tr>
</table>

The Command Set Type is 16-bit numeric value, used to identify a particular set of vendor defined commands supported by the Endpoint. This is described in the MCPT base specification and used for identifying vendor unique command sets.

6.1 Attestation Protocol Format

The messages from PA-RoT to AC-RoT will have the following fields

Field Name	Description
IC	(MCTP integrity check bit) Indicates whether the MCTP message is covered by an overall MCTP message payload integrity check
Message Type	Indicates MCTP Vendor defined message
MCTP PCI Vendor	Id for PCI Vendor
Rq	Request bit. This bit is used to help differentiate between MCTP control Request messages and other message classes.
D-bit	This bit is used to indicate whether the Instance ID field is being used for tracking and matching requests and responses
Crypt	Message body including Sequence No is encrypted
Sequence No	The sequence field is used to identify new instances of messages. This is used to match up response messages with the request.
Command	The command ID for command to execute
Msg Integrity Check	This field represents the optional presence of a message type-specific integrity check over the contents of the message body. If present (indicated by IC bit) the Message integrity check field is carried in the last bytes of the message body

Table 6 MCTP Message Format

<table>
<tr><td colspan="8">+0</td><td colspan="8">+1</td><td colspan="8">+2</td><td colspan="8">+3</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="4">MCTP Rsvd</td><td colspan="4">Header Version</td><td colspan="8">Destination Endpoint ID</td><td colspan="8">Source Endpoint ID</td><td>SOM</td><td>EOM</td><td colspan="2">Pkt Seq #</td><td>TO</td><td colspan="3">Msg Tag</td></tr>
<tr><td>IC</td><td colspan="7">Msg Type = 7E</td><td colspan="16">MCTP PCI Vendor ID</td><td>Rq</td><td>D</td><td>crypt</td><td colspan="5">Seq #</td></tr>
<tr><td colspan="8">Command</td><td colspan="24">Message Body</td></tr>
</table>

The protocol header fields are to be included only in the first packet of a multiple packet MCTP message. After reconstruction of the message body, the protocol header will be used to interpret the message contents.

Additionally, the Cerberus attestation protocol format also includes an 8-bit cyclic redundancy check (CRC-8) checksum at the end of every attestation protocol packet, for the entire MCTP packet including the MCTP header. This is independent of the message integrity check defined by the MCTP protocol.

6.2 Command Set Type Code

The type codes associated with the commands determine whether the command can be executed outside of an obfuscated session:

Table 7 Command Types

Type	Description
1	Accepted inside or outside session. Typically, pre-session commands
2	Authentication and session setup commands.
3	Session required commands, obfuscated by session encryption or KDF, message body content is normally scrambled.
8xh	Any of the other command types, but the command uses the timeout allowed for Cryptographic commands.

6.3 RoT Commands

The following table describes the commands defined under this specification There are two categories: (1) PA/AC for commands applicable to the PA-RoT and AC-RoT. (2) Commands applicable to the PA-RoT of Trust only. All MCTP commands are master initiated. The following section describes the command codes

Table 8 Command List

Register Name	Type	Command	R/W	RoT	Description
ERROR	01h	7Fh	R	PA/AC	ERROR Response message as per USB Type C Authentication specification.
Firmware Version	01h	01h	R/W	PA/AC	Retrieve firmware version information
Device Capabilities	01h	02h	R	PA/AC	Retrieves Device Capabilities
Device Id	01h	03h	R	PA/AC	Retrieves Device Id
Device Information	01h	04h	R	PA/AC	Retrieves device information
Export Certificate	01h	20h	R	PA/AC	Exports CSR
Import Certificate	81h	21h	W	PA/AC	Imports CA signed Certificate
Get Certificate State	01h	22h	R	PA/AC	Checks the state of the signed Certificate chain
GET_DIGESTS	82h	81h	R	PA/AC	PA-RoT retrieves session information
GET_CERTIFICATE	02h	82h	W	PA/AC	PA-RoT sets session variables based on Session Query
CHALLENGE	82h	83h	R	PA/AC	PA-RoT retrieves and verifies AC-RoT certificate
Key Exchange	82h	84h	W	PA	Exchange pre-master session keys and mfg device pairing key
Get Log Info	03h	4Fh	R	PA	Get Log Information
Get Log	03h	50h	R	PA	Retrieve debug, attestation and tamper log
Clear Debug Log	03h	51h	W	PA	Clear debug log
Get Host State	03h	40h	R	PA/AC	Get reset state of the host processor
Get PFM Id	03h	59h	R	PA	Get PFM Information
Get PFM Supported	03h	5Ah	R	PA	Retrieve the PFM
Prepare PFM	03h	5Bh	W	PA	Prepare PFM payload on PA-RoT
Update PFM	03h	5Ch	W	PA	Set the PFM
Activate PFM	03h	5Dh	W	PA	Force Activation of supplied PFM
Get CFM Id	03h	5Eh	R	PA	Get Component Manifest Id
Prepare CFM	03h	5Fh	W	PA	Prepare Component Manifest Update
Update CFM	03h	60h	W	PA	Update Component Manifest
Activate CFM	03h	61h	W	PA	Activate Component Firmware Manifest Update
Get CFM Supported	03h	93h	W	PA	Retrieve supported CFM IDs
Get PCD Id	03h	62h	R	PA	Get Platform Configuration Data Id
Prepare PCD	03h	63h	W	PA	Prepare Platform Configuration Data Update

Update PCD	03h	64h	W	PA	Update Platform Configuration Data
Activate PCD	03h	65h	R	PA	Activate Platform Configuration Data Update
Prepare Firmware Update	03h	66h	W	PA	Prepare for receiving firmware image
Update Firmware	03h	67h	W	PA	Firmware update payload
Update Status	03h	68h	W	PA	PFM/CFM/PCD Status
Activate Firmware Update	03h	69h	W	PA	Activate received FW update
Reset Configuration	83h	6Ah	R/W	PA/AC	Reset configuration to default state
Get Config IDs	83h	70h	R	PA/AC	Get manifest IDs and signed digest of request nonce and response ids.
Recovery Firmware	03h	71h	W	PA/AC	Restore Firmware Index using backup.
Prepare Recovery Firmware	03h	72h	W	PA	Prepare storage for Recovery Image
Update Recovery Firmware	03h	73h	W	PA	Updates the Recover image
Activate Recovery Firmware	03h	74h	W	PA	Activate the received Recovery image
Platform Configuration Register	83h	80h	R	PA	Returns the Platform Measurement
Extend Platform Configuration Register	83h	86h	R	PA	Extents Platform Measurements
Reset Counter		87h	R	PA	Reset Counter
Unseal Message	83h	89h	W	PA/AC	Unseal attestation challenges.
Unseal Message Result	03h	90h	W	PA/AC	Get unsealing status and result

6.4 Message Body Structures

The following section describes the structures of the MCTP message body.

6.5 Error Message

The error command is returned to on command responses when the command was not completed as proposed, it also acts as a generic status for commands without response whereby “No Error” code would indicate success. The Msg Tag, Seq and Command match the response to the corresponding request. The Message Body is returned as follows:

Table 9 Error Response

Payload	Description
1	Error Code
2:5	Error Data

Table 10 Error Codes

Error Code	Value	Description	Data
No Error	0h	Success [Reserved in USB Type C Authentication Specification]	00h

Invalid Request	01h	Invalidated data in the request	00h
Busy	03h	Device cannot response as it is busy processing other commands	00h
Unspecified	04h	Unspecified error occurred	Vendor defined
Reserved	05h-EFh	Reserved	Reserved
Invalid Checksum	F0h	Invalid checksum	Checksum
Out of Order Message	F1h	EOM before SOM	00h
Authentication	F2h	Authentication not established	00h
Out of Sequence Window	F2h	Message received out of Sequence Window	00h

If a response is undefined in the commands following section, the Error message is the expected with "No Error". The Error Message occurs on all response messages that fail for a given reason. or an error occurs in the response.

6.6 Firmware Version

This command gets the target firmware the version.

Table 11 Firmware Version Request

Payload	Description
1	Area Index: 00h = Entire Firmware 01h = RIoT Core Additional indexes are firmware specific

Table 12 Firmware Version Response

Payload	Description
1:32	Firmware Version Number ASCII Formatted

6.7 Device Capabilities

Device Capabilities provides information on device functionality. The minimum payload size negotiated is selected in communication.

Table 13 Device Capabilities Request

Payload	Description
1:2	Maximum Payload Size
3	Mode: [7:6] 00 = AC-RoT 01 = PA-RoT [5:4] Master/Slave 00 = Unknown 01 = Master

	10 = Slave 11 = both master and slave [3] Reserved [2:0] Security 000 = None 001 = Hash/KDF 010 = Authentication [Certificate Auth] 100 = Confidentiality [AES]
4	[7] PFM support [6] Policy Support [5] Firmware Protection [4-0] Reserved
5	PK Key Strength: [7] RSA [6] ECDSA [5:3] ECC 000: None 001: 160bit 010: 256bit 100: Reserved [2:0] RSA: 000: None 001: RSA 2048 010: RSA 3072 100: RSA 4096
6	Encryption Key Strength: [7] ECC [6:3] Reserved [2:0] AES: 000: None 001: 128 bit 010: 256 bit 100: 384 bit

Table 14 Device Capabilities Response

Payload	Description
1:2	Max payload size. Describes the maximum payload the device can accept.
3	Mode: [7:6] 00 = AC-RoT 01 = PA-RoT [5:4] Master/Slave 00 = Unknown 01 = Master 10 = Slave 11 = both master and slave [3] Reserved

	[2:0] Security 000 = None 001 = Hash/KDF 010 = Authentication [Certificate Auth] 100 = Confidentiality [AES]
4	[7] PFM support [6] Policy Support [5] Firmware Protection [4-0] Reserved
5	PK Key Strength: [7] RSA [6] ECDSA [5:3] ECC 000: None 001: 160bit 010: 256bit 100: Reserved [2:0] RSA: 000: None 001: RSA 2048 010: RSA 3072 100: RSA 4096
6	Encryption Key Strength: [7] ECC [6:3] Reserved [2:0] AES: 000: None 001: 128 bit 010: 256 bit 100: 384 bit
7:8	Response Delay Byte 1 Message: Maximum timeout: x (multiples of 10ms) Byte 2 Cryptographic: Maximum Signature: X (multiple of 100ms)

6.8 Device Id

Eight bytes response.

Table 15 Device Id Request

Payload	Description

Table 16 Device Id Response

Payload	Description
1:2	Vendor ID; LSB

3:4	Device ID; LSB
5:6	Subsystem Vendor ID; LSB
7:8	Subsystem ID; LSB

6.9 Device Information

This command gets information about the target device.

Table 17 Device Information Request

Payload	Description
1	Information Index: 00h = Unique Chip Identifier Additional indexes are firmware specific

Table 18 Device Information Response

Payload	Description
1:N	Requested information in binary format

6.10 Export CSR

Exports the Device Identification Certificate Self Signed Certificate Signing Request. The initial Certificate is self-signed, until CA signed and imported. Once the CA signed version of the certificate has been imported to the device, the self-signed certificate is replaced.

Table 19 Export CSR Request

Payload	Description
1	Index: Default = 0

Table 20 Export CSR Response

Payload	Description
1:N	Certificate

6.11 Import Certificate

Imports the signed certificate into the device. Export and Import Indexes must match. Upon verification, the device is sealed an no further imports can occur without changes to firmware.

Table 21 Import Certificate Request

Payload	Description
1	Index: Default = 0
2:3	Certificate Length
4:N	Certificate

6.12 Get Certificate State

Determine the state of the certificate chain for signed certificates that have been sent to the device. The request for this command contains no additional payload.

Table 22 Get Certificate State Response

Payload	Description
1	State: 0 = A valid chain has been provisioned. 1 = A valid chain has not been provisioned. 2 = The stored chain is being validated.
2:4	Error details if chain validation has failed.

6.13 GET DIGEST

This command closely matches the USB Type C-Authentication Message. The Protocol Version byte is not included, the Message Type is present in the Command byte of the MCTP message. See: Table 6 MCTP Message Format.

This command can be sent at any time. If authentication was previously established, this command will renegotiate/override the previous authentication and session establishment. The byte 2, reserved USB Type C – authentication specification, is repurposed to describe the key exchange algorithm. This is relevant when the requester and responder support multiple key exchange algorithms.

Table 23 USB Type C – Authentication GET DIGEST Request

Payload	Description
1	Param1 - Reserved
2	Key Exchange Algorithm: 0 = RSA 1 = ECDHE

Table 24 USB Type C – Authentication GET DIGEST Response

Payload	Description
1	Capabilities Field; shall be set to 01
2	The number of digests returned shall be equal to the number of bits set in this byte. The digests shall be returned in order of increasing slot number. This byte can be used to determine the payload size.
3:N	Digest[0] 32 byte SHA256 digest of the first Certificate in the Chain
N+	Digest[1] 32 byte SHA256 digest of N Certificate in the Chain

6.14 GET_CERTIFICATE

This command retrieves the public attestation certificate chain for the AC-RoT. It follows closely the USB Type C Authentication Specification.

Table 25 USB Type C – Authentication CERTIFICATE Request

Payload	Description
1	Param1: Slot Number of the target Certificate Chain to read. The value should be 0-7.
2	Certificate number
3:4	Offset: offset in bytes from start of the Certificate chain where read request begins.
5:6	Length: number of bytes to read

Table 26 USB Type C – Authentication CERTIFICATE Response

Payload	Description
1	Param1: Slot Number of the target Certificate Chain returned.
2	Certificate number
3:N	Requested contents of target Certificate Chain. See USB Type-C Authentication Specification for further details.

6.15 CHALLENGE

The PA-RoT will send this command providing the first nonce in the key exchange.

Table 27 CHALLENGE_AUTH Request

Payload	Description
1	Slot number of the recipient's Certificate Chain that will be used for Authentication
2	Reserved
3:35	Random 32 byte nonce chosen by PA-RoT

Table 28 CHALLENGE_AUTH Response

Payload	Description
1	Shall contain the Slot number in the Param1 field of the corresponding CHALLENGE Request
2	Certificate slot mask
3	MinProtocolVersion supported by device
4	MaxProtocolVersion supported by device
5	Capabilities Set to 01h
6	Reserved
7	CertChainHash 32-byte SHA256 hash of the Certificate Chain used for Authentication
39	Salt Random number chosen by AC-RoT (^{RN2})
40:(L)71	Value of Platform Measurement Register 0 (Aggregated Firmware Digest)
103 + SNG	Signature of combined request and response message payloads. See USB Type C Authentication Protocol for details of request/response signature.

The firmware digests are measurements of the security descriptors and the firmware of the target components. This firmware measurement data does not include Cerberus PCD, CFM or PFM. These measurements are returned in PMR2. The numbers are retrieved in the 6.41 Get Configuration Ids. The USB 3.0 Context Hash is not included in the CHALLENGE_AUTH. This is replaced with contextual measurements for the device. Note: The attestation Certificate derivation will include the measurement of firmware and security descriptors. PMR0 is anticipated to be the least changing PMR as it contains the measurement of security descriptors and device initial boot loader.

6.16 Key Exchange

Key exchange is used by the PA-RoT and caller to create an encrypted channel. After verifying the Certificate authenticity, a pre-session key can be generated for establishing session confidentiality.

Table 29 Key Exchange Request

Payload	Description
1	Key Type: 0 = Session Key, 1 = Paired Key
2:N	key type = 0: (PK1)Encrypted pre-session key (RN3) or ephemeral ECDHE key. Value determined by advertised device capabilities. key type = 1: HMAC(Session key, Mfg Pair Key)

Table 30 Key Exchange Response

Payload	Description
1	See Error Code

Upon receiving this key with Key Type 0, both sides can establish an encrypted session. If Key Type 1, the paired key is compared and paired verified functionality unlocked. The Key Type 1 can only be performed under an encrypted session, as the session key is required for the HMAC.

6.17 Get Log Info

Get the internal log information for the RoT.

Table 31 Get Log Info Request

Payload	Description

Table 32 Get Log Info Response

Payload	Description
1:4	Debug Log (01h) Length in bytes
5:8	Attestation Log (02h) Length in bytes
9:12	Tamper Log (03h) Length in bytes

6.18 Get Log

Get the internal log for the RoT. There are 3 types of logs available: The Debug Log, which contains Cerberus application information and machine state. The Attestation measurement log, this log format is like the TCG log, and the Tamper log. It is not possible to clear or reset the tamper counter. Log formats are discussed in the Log specification.

Table 33 Log Types

Log Type	Description
1	Debug Log
2	Attestation Log
3	Tamper Log

Table 34 Get Log Section Request

Payload	Description
1	Log Type
2	Log Instance (default 0)
3:6	Offset

Table 35 Get Debug/Attestation Log Response

Payload	Description
1:N	The contents of the log

Length determined by end of log, or packet size based on device capabilities see section: 6.7 Device Capabilities. If response spans multiple MCTP messages, end of response will be determined by an MCTP packet which has payload less than maximum payload supported by both devices. To guarantee a response will never fall exactly on the max payload boundary, the responder should send back an extra packet with zero payload.

6.19 Clear Debug/Attestation Log

Clear the log in the RoT.

Table 36 Clear Debug/Attestation Log Request

Payload	Description
1	Type: 01 or 02

Note: in clearing the attestation log, it is automatically recreated using current measurements.

6.20 Get Host State

Retrieve the reset state of the host processor being protected by Cerberus.

Table 37 Get Host State Request

Payload	Description
1	Port: 0 - 1

Table 38 Get Host State Response

Payload	Description
1	Host Reset State: 00h – Host is running (out of reset) 01h – Host is being held in reset 02h – Host is not being held in reset, but is not running

6.21 Get Platform Firmware Manifest Id

Retrieves the PFM Id

Table 39 PFM Information Request

Payload	Description
1	Port: 0 -1
2	Active: 0 Pending: 1

Table 40 PFM Id Response

Payload	Description
1	PFM Valid (0 or 1)
2:5	PFM Id

6.22 Get Platform Firmware Manifest Supported Firmware

Table 41 PFM Supported Firmware Request

Payload	Description
1	Port: 0 -1
2	Active: 0 Pending: 1
3:6	Offset

Table 42 Supported Firmware Response

Payload	Description
1	PFM Valid (0 or 1)
2:5	PFM ID
6:N	PFM supported FW versions

If response spans multiple MCTP messages, end of response will be determined by an MCTP packet which has payload less than maximum payload supported by both devices. To guarantee a response will never fall exactly on the max payload boundary, the responder should send back an extra packet with zero payload.

6.23 Prepare Platform Firmware Manifest

Provisions RoT for incoming PFM.

Payload	Description
1	Port Id
2:5	Total size

6.24 Update Platform Firmware Manifest

The flash descriptor structure describes the regions of flash for the device.

Table 43 Update PFM Request

Payload	Description
1	PortId
2:N	PFM Payload

PFM payload includes PFM signature and monotonic forward only Id. PFM signature is verified upon receipt of all PFM payloads.   PFMs are activated upon the activation command.   Note if a system is rebooted after receiving a PFM, the PFM is atomically activated.   To activate before reboot, issue the Activate PFM command.

6.25 Activate Platform Firmware Manifest

Upon valid PFM update, the update command seals the PFM committal method. If committing immediately, flash reads and writes should be suspended when this command is issued. The RoT will master the SPI bus and verify the newly updated PFM. This command can only follow a valid PFM update.

Table 44 Update PFM Request

Payload	Description
1	PortId
2	Activation: 0 = Reboot only 1 = Immediately

If reboot only has been issued, the option for “Immediately” committing the PFM is not available until a new PFM is updated.

6.26 Get Component Firmware Manifest Id

Retrieves the Component Firmware Manifest Id

Table 45 Get CFM Request

Payload	Description
1	CFM region (active 0, pending 1)

Table 46 CFM Response

Payload	Description
1	CFM Valid (0 or 1)
2:5	CFM Id

6.27 Prepare Component Firmware Manifest

Provisions RoT for incoming Component Firmware Manifest.

Payload	Description
1:4	Total size

6.28 Update Component Firmware Manifest

The flash descriptor structure describes the regions of flash for the device.

Table 47 Update Component Firmware Manifest Request

Payload	Description
1:N	Component Firmware Manifest Payload

The CFM payload includes CFM signature and monotonic forward only Id. CFM signature is verified upon receipt of all CFM payloads. CFMs are activated upon the activation command. Note if a system is rebooted after receiving a CFM, the pending CFM is verified and atomically activated. To activate before reboot, issue the Activate CFM command.

6.29 Activate Component Firmware Manifest

Upon valid CFM update, the update command seals the CFM committal method. The RoT will master I2C and attest Components in the Platform Configuration Data against the CFM.

Table 48 Active CFM Request

Payload	Description
	Activation: 0 = Reboot only 1 = Immediately

6.30 Get Component Firmware Manifest Component IDs

CFM Supported component IDs Request

Payload	Description

1	Active: 0 Pending: 1
2:5	Offset

CFM Supported component IDs Response

Payload	Description
1	CFM Valid (0 or 1)
2:5	CFM ID
6:N	CFM supported component IDs

If response spans multiple MCTP messages, end of response will be determined by an MCTP packet which has payload less than maximum payload supported by both devices. To guarantee a response will never fall exactly on the max payload boundary, the responder should send back an extra packet with zero payload.

6.31 Get Platform Configuration Data Id

Retrieves the PCD Id

Table 49 Get Platform Configuration Data Request

Payload	Description

Table 50 Get Platform Configuration Data Response

Payload	Description
1	PCD Valid (0 or 1)
2:5	PCD ID

6.32 Prepare Platform Configuration Data

Provisions RoT for incoming Platform Configuration Data.

Payload	Description
1:4	Total size

6.33 Update Platform Configuration Data

The flash descriptor structure describes the regions of flash for the device.

Table 51 Update Platform Configuration Data Request

Payload	Description
1:N	PCD Payload

The PCD payload includes PCD signature and monotonic forward only Id. PCD signature is verified upon receipt of all PCD payloads. PCD is activated upon the activation command. Note if a system is rebooted after receiving a PCD.

6.34 Activate Platform Configuration Data

Upon valid PCD update, the activate command seals the PCD committal.

Table 52 Active PCD Request

Payload	Description

6.35 Platform Configuration

The following table describes the Platform Configuration Data Structure

Table 53 PCD Structure

<table>
<tr><th>Payload</th><th>Description</th></tr>
<tr><td>1:3</td><td>Platform Configuration Data Id</td></tr>
<tr><td>4:5</td><td>Length</td></tr>
<tr><td>6</td><td>Policy Count</td></tr>
<tr><td>7:N</td><td>Each AC-RoT has 1 entry. The Configuration Data determines the feature enablement and attestation
<table>
<tr><td>Byte</td><td>Description</td></tr>
<tr><td>1</td><td>Device Id</td></tr>
<tr><td>4</td><td>Channel</td></tr>
<tr><td>5</td><td>Slave Address</td></tr>
<tr><td>6</td><td>[7:5] Threshold Count
[4] Power Control
0 = Disabled
1 = Enabled
[3] Debug Enabled
0 = Disabled
1 = Enabled
[2] Auto Recovery
0 = Disabled
1 = Enabled
[1] Policy Active
0 = Disabled
1 = Enabled
[0] Threshold Active
0 = Disabled
1 = Enabled</td></tr>
<tr><td>7</td><td>Power Ctrl Index</td></tr>
<tr><td>8</td><td>Failure Action</td></tr>
</table>
</td></tr>
</table>

N:N	Signature of payload

The Power Control Index informs the PA-RoT of the index assigned to power sequence the Component. This informs the PA-RoT which control register needs to be asserted in the platform power sequencer.

The Failure Action: 0 = Platform Defined, 1 = Report Only, 2 = Auto Recover 3 = Power Control.

6.36 Prepare Firmware Update

Provisions RoT for incoming firmware update.

Table 54 Prepare Firmware Update

Payload	Description
1:4	Total size

6.37 Update Firmware

The flash descriptor structure describes the regions of flash for the device.

Table 55 Update Firmware Request

Payload	Description
1:N	Firmware Update payload, header signature. See firmware update specification.

6.38 Update Status

The Update Status reports the update payload status. The update status will be status for the last operation that was requested. This status will remain the same until another operation is performed or Cerberus is reset.

Payload	Description
1	Update Type 00 = Firmware 01 = Platform Firmware Manifest 02 = Component Firmware Manifest 03 = Configuration Data 04 = Host Firmware 05 = Recovery Firmware 06 = Reset Configuration
2	Port

Table 56 Update Status Response Request

Payload	Description
1:4	Update Status. See firmware update specification for details.

6.39 Activate Firmware Update

Alerts Cerberus that sending of update bytes is complete, and that verification of update should start. This command has no payload, the ERROR response zero is expected.

Table 57 Activate Firmware Update

Payload	Description

Table 58 Activate Firmware Update

Payload	Description
1	ERROR CODE

6.40 Reset Configuration

Resets configuration parameters back to the default state. Depending on the request parameters, different amounts of types of configuration can be erased, and each type of configuration may require different levels of authorization to complete.

If authorization is required for the operation to complete, the response will contain a device-specific, one-time use, authorization token that must be signed with the PFM key to unlock the operation. The authorization token has the following behavior:

1. A request for the same operation without providing the signed authorization token will generate a new token that invalidates any old token. This is true even if the old token has not been used yet.
2. After an authorization token has been used to unlock an operation, it can never be used again. A new token must be requested. This is true even if the requested operation was not able to complete successfully.
3. A failure to authorize the request when providing a signed token does not invalidate the current authorization token in the device.

If authorization is not required, or the request is sent with a signed token, a standard error response will be returned indicating the status.

Table 59 Reset Configuration Request

Payload	Description
1	Type of reset operation to request: 0: Revert the device into the unprotected (bypass) state by erasing all PFMs and CFMs.

	1: Perform a factory reset by removing all configuration. This does not include signed device certificates.
2:N	(Optional) Device-specific authorization token, signed with PFM key.

Table 60 Reset Configuration Response with Authorization Token

Payload	Description
1:N	Device-specific authorization token

6.41 Get Configuration Ids

This command retrieves PFM Ids, CFM Id, PCD Id, and signed digest of request nonce and response ids.

Table 61 Get Configuration Ids Request

Payload	Description
1:16	128bit Nonce

Table 62 Get Configuration Ids Response

Payload	Description
1:16	128bit Nonce
17	Number of PFMs Ids (P)
18	Number of CFM Ids (C)
19 : (P*4 + C *4 + 4)	PMF Id[0] - PFM Id[N] CMF Id[0] - CFM Id[N] PCD
20 + (P*4 + C *4 + 4) + SGN	SGN$^{(pk)}$(request message nonce + response message payload)

N is the number of measurements and L is the length of each measurement. The Signature should be a SHA2 over the request and response message body (excluding the signature 5+L*N). The signature algorithm is defined by the certificate exchanged in the DIGEST.

6.42 Recover Firmware

Start the firmware recovery process for the device. Not all devices will support all types of recovery. The implementation is device specific.

Table 63 Recover Firmware Request

Payload	Description
1	Port Index
2	Firmware image to use for recovery:

	0: Exit Recovery 1: Enter Recovery

6.43 Prepare Recovery Image

Provisions RoT for incoming Recovery Image for Port.

Table 64 Prepare Recovery Image Request

Payload	Description
1	Port
2:5	Total size

The response back is the Error Code indicating Success or failure.

6.44 Update Recovery Image

The flash descriptor structure describes the regions of flash for the device.

Table 65 Update Component Firmware Manifest Request

Payload	Description
1	Port
2:N	Recovery Image Payload

6.45 Activate Recovery Image

Signals recovery image has been completely sent and verification of the image should start. Once the image has been verified, it can be used for host firmware recovery.

Table 66 Activate Recovery Image

Payload	Description
1	Port

Table 67 Activate Recovery Image

Payload	Description
1	ERROR CODE

6.46 Get Recovery Image Version Id

Retrieves the recovery image version Id.

Table 68 Recovery Image Information Request

Payload	Description
1	Port

Table 69 Recovery Image Information Response

Payload	Description
1:32	Recovery Image Version Id

6.47 Platform Measurement Register

Returns the Cerberus Platform Measurement Register (PMR), which is a digest of the Cerberus Firmware, PFM, CFM and PCD. This information contained in PMR0 is Cerberus firmware. PMR1-2 are reserved for Filter Ports 1 and Ports 2 PFM/CFM. PMR3-4 are reserved for external usages. Attestation Log 0 will provide PMR log. Attestation Log N will provide PMR1-4.

Table 70 Platform Measurement Request

Payload	Description
1	Platform Measurement Number
2:17	128bit Nonce

Table 71 Platform Measurement Response

Payload	Description
1:32	Platform Measurement Value
33:N	SGN$^{(pk)}$(request message payload + response message payload)

PMR1-4 are cleared on component reset. PMR0 is cleared and re-built on Cerberus reset.

6.48 Update Platform Measurement Register

External updates to PMR3-4 are permitted. Attempts to update PMR0-2 will result error. Only SHA 2 is support for measurement extension. SHA1 and SHA3 are not applicable. Note: The measurement can only be updated over an authenticated and secured channel.

Table 72 Update Platform Measurement Request

Payload	Description
1	Platform Measurement Number
2:33	Measurement Extension

Table 73 Update Platform Measurement Response

Payload	Description
1	ERROR CODE

6.49 Reset Counter

Provides Cerberus and Component Reset Counter since power-on.

Table 74 Reset Counter Request

Payload	Description
1	Reset Counter Type
2	Port Id

Table 75 Reset Counter Response

Payload	Description
1:2	Reset Count

6.50 Message Unseal

This command starts unsealing an attestation message.

Table 76 Unseal Message Request

Payload	Description
1:2	Seed Length (S)
2:S+2	Seed
S+2:S+4	Cipher Text Length (C)
S+4:S+C+4	Cipher Text
S+C+6:S+C+38	HMAC
S+C+38:S+C+102	Sealing

6.51 Message Unseal Result

This command retrieves current status of an unsealing process.

Table 77 Unseal Message Request

Payload	Description

Table 78 Unseal Message Response

Payload	Description
1:4	Unsealing status
5:6	Encryption Key Length
7:N	Encryption Key

The Seal/Unseal flow is described in the Cerberus Attestation Integration specification.

7 Platform Active RoT (PA-RoT)

The PA-RoT is responsible for challenging the AC-RoT's and collecting their firmware measurements. The PA-RoT retains a private manifest of active components that includes addresses, buses, firmware versions, digests and firmware topologies.

The manifest informs the PA-RoT on all the Active Components in the system. It provides their I2C addresses, and information on how to verify their measurements against a known or expected state. Polices configured in the Platform RoT determine what action it should take should the measurements fail verification.

In the Cerberus designed motherboard, the PA-RoT orchestrates power-on. Only Active Components listed in the challenge manifest, that pass verification will be released from power-on reset.

7.1 Platform Firmware Manifest (PFM) and Component Firmware Manifest

The PA-RoT contains a Platform Firmware Manifest (PFM) that describes the firmware permitted on the Platform. The Component Firmware Manifest (CFM) describes the firmware permitted for components in the Platform. The Platform Configuration Data (PCD), specific to each SKU describes the number of Component types in the platform and their respective locations.

Note: The PFM and CFM are different from the boot key manifest described in the Processor Secure Boot Requirements specification. The PFM and CFM describe firmware permitted to run in the system across Platform and Active Components. The CFM is a complement to the PFM, generated by the PA-RoT for the measurement comparison of components in the system. This complement is as the Reported Firmware Manifest (RFM), which is like the TCG log. The PFM and RFM are stored encrypted in the PA-RoT. The symmetric encryption key for the PA-RoT is hardware generated and unique to each microcontroller. The symmetric key in the PA-Rot is not exportable or firmware readable; and only accessible to the crypto engine for encryption/decryption. The AES Galois/Counter Mode (GCM) encryption a unique auditable tag to any changes to the manifest at both an application level and persistent storage level.

The following table lists the attributes stored in the PFM for each Active component:

Table 79 PFM Attributes

Attribute	Description
Description	Device Part or Description
Device Type	Underlying Device Type of AC-RoT
Remediation Policy	Policy(s) defining default remediation actions for integrity failure.
Firmware Version	List of firmware versions
Flash Areas/Offsets	List of offset and digests, used and unused
Measurement	Firmware Measurements
Measurement Algorithm	Algorithm used to calculate measurement.

Public Key	Public keys in the key manifest
Digest Algorithm	Algorithm used to calculate
Signature	Firmware signature(s)

The PA-RoT actively takes measurements of flash from platform firmware, the PFM provides metadata that instructs the RoT on measurement and signature verification. The PA-RoT stores the measurements in the RFM. The PA-Rot then challenges the AC-RoTs for their measurements using the Platform Configuration Data. it compares measurements from the AC-RoT's to the CFM, while recording measurements in the RFM.

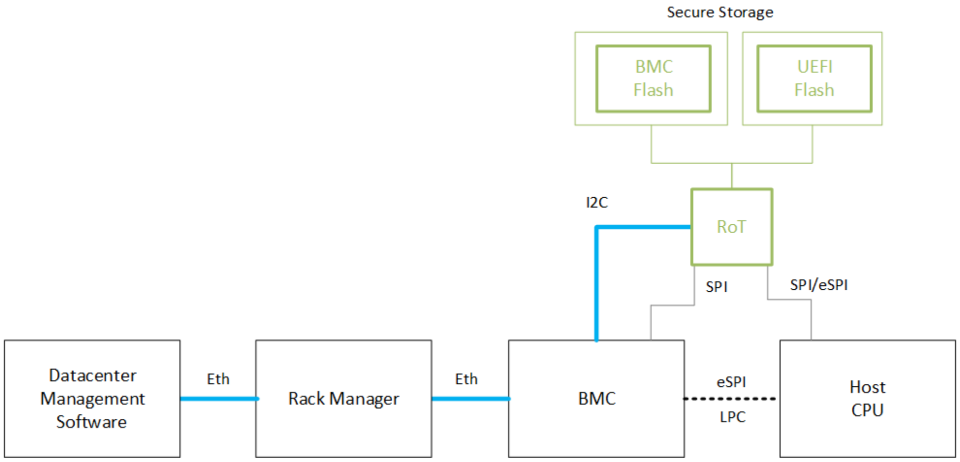
The measurements of the Platform firmware and Component firmware are compared to the PFM and CFM. Should a mismatch occur, the PA-RoT would raise an event log and invoke the policy action defined for the Platform and/or Component. A variety of actions can be automated for a PFM/CFM challenge failure. Actions are defined in the CFM and PCD files.

Note: The PA-RoT and AC-RoT enforce secure boot and only permit the download of digitally signed and unrevoked firmware. A PFM or CFM mismatch can only occur when firmware integrity is brought into question.

7.2 RoT External Communication interface

The PA-RoT connects to the platform through, either SPI, QSPI depending on the motherboard. Although the PA-RoT physically connects to the SPI bus, the microprocessor appears transparent to the host as it presents only a flash interface. The management interface into the PA-RoT and AC-RoTs is an I2C bus channeled through the Baseboard Management Controller (BMC). The BMC can reach all AC-RoTs in the platform. The BMC bridges the PA-RoT to the Rack Manager, which in-turn bridges the rack to the Datacenter management network. The interface into the PA-RoT is as follows:

Figure 11 External Communication Interface

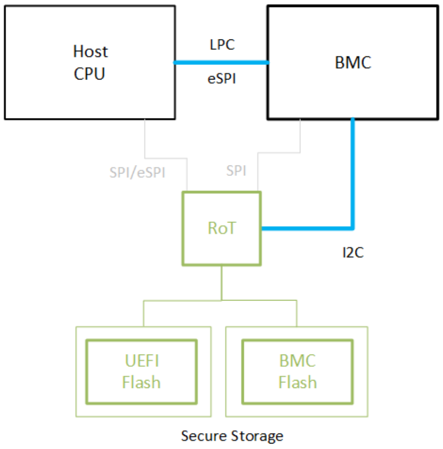


The Datacenter Management (DCM) software can communicate with the PA-RoT Out-Of-Band (OOB) through the Rack Manager. The Rack Manager allows tunneling through to the Baseboard Management Controller, which connects to the PA-RoT over I2C. This channel is assumed insecure, which is why all communicates are authenticated and encrypted. The Datacenter Management Software can collect the RFM measurements and other challenge data over this secure channel. Secure updates are also possible over this channel.

7.3 Host Interface

The host can communicate with the PA-RoT and AC-RoTs through the BMC host interface. Similar to the OOB path, the BMC bridges the host-side LPC/eSPI interface to the I2C interface on the RoT. The host through BMC is an unsecure channel, and therefore requires authentication and confidentiality.

Figure 12 Host Interface



7.4 Out Of Band (OOB) Interface

The OOB interface is essential for reporting potential firmware compromises during power-on. Should firmware corruption occur during power-on, the OOB channel can communicate with the DCM software while the CPU is held in reset. If the recovery policy determines the system should remain powered off, it's still possible for the DCM software to interrogate the PA-RoT for detailed status and make a determination on the remediation.

The OOB communication to Cerberus requires TLS and Certificate Authentication.

8 Legacy Interface

The legacy interface is defined for backward combability with devices that do not support MCTP. These devices must provide a register set with specific offsets for Device Capabilities, Receiving Alias Certificate, accepting a Nonce, and providing an offset for Signed Firmware Measurements. The payload structures will closely match that of the MCTP protocol version. Legacy interfaces to no support session based authentication but permit signed measurements.

8.1 Protocol Format

The legacy protocol leverages the SMBus Write/Read Word and Block commands. The interface is register based using similar read and write subroutines of I2C devices. The data transmit and receive requirements are 32 bytes or greater. Large payloads can be truncated and retrieved recursively spanning multiple block read or write commands.

The block read SMBUS command is specified in the SMBUS specification. Slave address write and command code bytes are transmitted by the master, then a repeated start and finally a slave address read. The master keeps clocking as the slaves responds with the selected data. The command code byte can be considered register space.

8.2 PEC Handling

An SMBus legacy protocol implementation may leverage the 8bit SMBus Packet Error Check (PEC) for transactional data integrity. The PEC is calculated by both the transmitter and receiver of each packet using the 8-bit cyclic redundancy check (CRC-8) of both read or write bus transaction. The PEC accumulates all bytes sent or received after the start condition.

An Active RoT that receives an invalid PEC can optionally NACK the byte that carried the incorrect PEC value or drop the data for the transaction and any further transactions (read or write) until the next valid read or write Start transaction is received.

8.3 Message Splitting

The protocol supports Write Block and Read Block commands. Standard SMBus transactions are limited to 32 bytes of data. It is expected that some Active Component RoTs with intrinsic Cerberus capabilities may have limited I2C message buffer designed around the SMBus protocol that limit them to 32 bytes. To overcome hardware limitations in message lengths, the Capabilities register includes a buffer size for determining the maximum packet size for messages. This allows the Platform's Active RoT to send messages larger than 32 bytes. If the Active Component RoT only permits 32 bytes of data, the Platform's Active RoT can segment the Read or Write Blocks into multiple packets totaling the entire message. Each segment includes decrementing packet number that sequentially identifies the part of the overall message. To stay within the protocol length each message segment must be no longer than 255 bytes.

8.4 Payload Format

The payload portions of the SMBus Write and Read blocks will encapsulate the protocol defined in this specification. The SMBus START and STOP framing and ACK/NACK bit conditions are omitted from this portion of the specification for simplification. To review the specifics of START and STOP packet framing and ACK/NACK conditions refer to the SMBus specification.

The data blocks of the Write and Read commands will encapsulate the message payload. The encapsulated payload includes a uint16 register offset and data section.

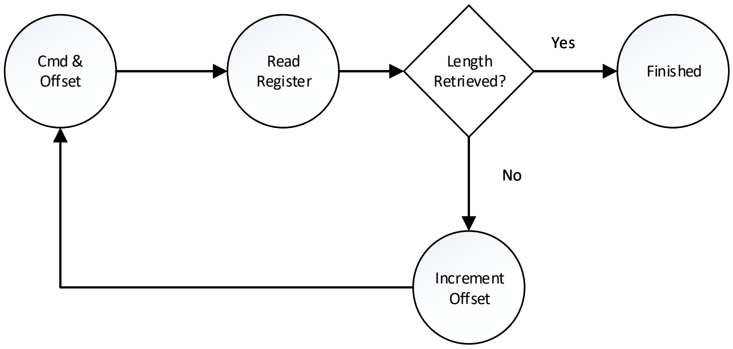
8.5 Register Format

The SMBUS command byte indexes the register, while additional writes offsets index inside the register space. The offset and respective response is encapsulated into the data portions of I2C Write and Read Block commands. The PA-RoT is always the I2C master, therefore Write and Read commands are described from the perspective of the I2C master.

Certain registers may contain partial or temporary data while the register is being written across multiple commands. The completion or sealing of register writes can be performed by writing the seal register to the zero offset.

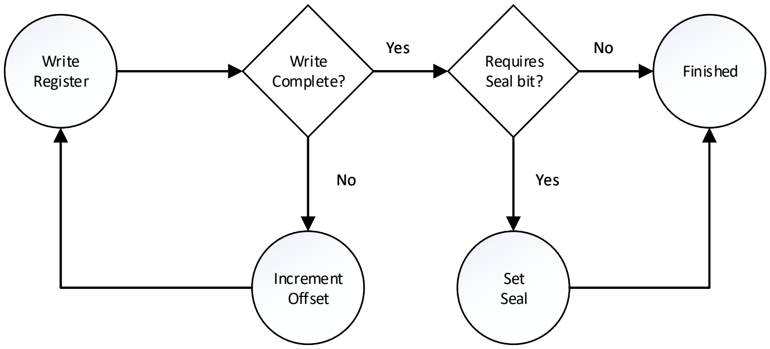
The following diagram depicts register read access flow for a large register space:

Figure 13 Register Read Flow



The following diagram depicts register write access flow for a large register space, with required seal (update complete bit):

Figure 14 Register Write Flow



8.6 Legacy Active Component RoT Commands

The following table describes the commands accepted by the Active Component RoT. All commands are master initiated. The command number is not representative of a contiguous memory space, but an index to the respective register

Table 80 Commands

Register Name	Command	Length	R/W	Description
Status	30h	2	R	Command Status
Firmware Version	32h	16	R/W	Retrieve firmware version information
Device Id	33h	8	R	Retrieves Device Id
Capabilities	34h	9	R	Retrieves Device Capabilities
Certificate Digest	3C	32	R	SHA256 of Device Id Certificate
Certificate	3D	4096	R/W	Certificate from the AC-Rot
Challenge	3E	32	W	Nonce written by RoT
Platform Configuration Register	03h	5Eh	R	Reads firmware measurement, calculated with S Nonce

8.7 Legacy Command Format

The following section describes the register format for AC-RoT that do not implement SMBUS and comply with the legacy measurement exchange protocol.

8.7.1 Status

The SMBUS read command reads detailed information on error status. The status register is issued between writing the challenge nonce and reading the Measurement. The delay time for deriving the Measurement must comply with the Capabilities command.

Table 81 Status Register

Payload	Description
1	Status: 00 = Complete 01 In Progress 02 Error
2	Error Data or Zero

8.7.2 Firmware Version

The SMBUS write command payload sets the index. The subsequent SMBUS read command reads the response. For register payload description see response: Table 12 Firmware Version Response

8.7.3 Device Id

The SMBUS read command reads the response. For register payload description see response: Table 1 Field Definitions

8.7.4 Device Capabilities

The SMBUS read command reads the response. For register payload description see response:

Table 14 Device Capabilities Response

8.7.5 Certificate Digest

The SMBUS read command reads the response. For register payload description see response: Table 24 USB Type C – Authentication GET DIGEST Response

The PA-Rot will use the digest to determine if it has the certificate already cached. Unlike MCTP, only the Alias and Device Id cert is supported. Therefore, it must be CA signed by a mutually trusted CA, as the CA Public Cert is not present

8.7.6 Certificate

The SMBUS write command writes the offset into the register space. For register payload description see response: Table 3 Certificate Chain

Unlike MCTP, only the Alias and Device Id cert is supported. Therefore, it must be CA signed by mutually trusted CA, as the CA Public Cert is not present in the reduced challenge

The SMBUS write command writes a nonce for measurement freshness.

Table 82 Challenge Register

Payload	Description
1:32	Random 32 byte nonce chosen by PA-RoT

8.7.7 Measurement

The SMBUS read command that reads the signed measurement with the nonce from the hallenge above. The PA-RoT must poll the Status register for completion after issuing the Challenge and before reading the Measurement.

Table 83 Measurement Register

Payload	Description
1	Length (L) of following hash digest.
2:33	H(Challenge Nonce || H(Firmware Measurement/PMR0))
34:N	Signature of HASH [2:33]

9 References

9.1 DICE Architecture

https://trustedcomputinggroup.org/work-groups/dice-architectures

9.2 RIoT

https://www.microsoft.com/en-us/research/publication/riot-a-foundation-for-trust-in-the-internet-of-things

9.3 DICE and RIoT Keys and Certificates

https://www.microsoft.com/en-us/research/publication/device-identity-dice-riot-keys-certificates

9.4 USB Type C Authentication Specification

http://www.usb.org/developers/docs/

9.5 PCIe Device Security Enhancements specification

https://www.intel.com/content/www/us/en/io/pci-express/pcie-device-security-enhancements-spec.html

9.6 NIST Special Publication 800-108

Recommendation for Key Derivation Using Pseudorandom Functions.

http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-108.pdf