



# Intel® TDX Connect TEE-IO Device Guide

---

May 2025

## Disclaimers

Intel Corporation ("Intel") provides these materials as-is, with no express or implied warranties.

All products, dates, and figures specified are preliminary, based on current expectations, and are subject to change without notice. Intel does not guarantee the availability of these interfaces in any future product. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described might contain design defects or errors known as errata, which might cause the product to deviate from published specifications. Current, characterized errata are available on request.

Intel technologies might require enabled hardware, software, or service activation. Some results have been estimated or simulated. Your costs and results might vary.

No product or component can be absolutely secure.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein.

No license (express, implied, by estoppel, or otherwise) to any intellectual-property rights is granted by this document.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice.

Copies of documents that have an order number and are referenced in this document or other Intel literature may be obtained by calling 1-800-548-4725 or by visiting <http://www.intel.com/design/literature.htm>.

© Intel Corporation, 2025. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands might be claimed as the property of others.

# Table of Contents

Disclaimers.....	2
Table of Contents.....	3
Introduction and Scope.....	6
Glossary .....	8
<b>1 TEE-IO Device Architecture Overview.....</b>	<b>10</b>
1.1. TEE-IO Security Model.....	10
1.2. Security Model for the Device.....	11
1.3. Implementing TEE-IO Device Stack .....	11
1.4. Secure Device Interface Lifecycle Example.....	12
<b>2 TEE-IO Software Stack.....</b>	<b>15</b>
2.1. Software Stack Overview.....	15
2.2. Device Communication (DOE).....	16
2.3. Device Attestation (SPDM) .....	16
2.4. Software Secure Communication (SPDM) .....	17
2.5. Link Encryption Key Management (IDE_KM) .....	18
2.6. Device Interface Management (TDISP) .....	19
2.7. Implementation Reference .....	21
2.7.1. SPDM software stack.....	21
2.7.2. IDE_KM software stack .....	22
2.7.3. TDISP software stack .....	22
<b>3 TEE-IO Hardware Stack.....</b>	<b>24</b>
3.1. IDE Stream .....	24
3.1.1. IDE Stream Precedence.....	27
3.1.2. TEE Limited Stream .....	27
3.1.3. Partial Header Encryption .....	27
3.2. TDI TLP Rule.....	28
3.2.1. TDI as Completer .....	30
3.2.2. TDI as Requester .....	33
3.2.3. ATS Rule .....	34
3.2.4. Peer to Peer (P2P).....	36

- 3.3. TDISP Interoperability with PCIe Capabilities ..... 37
  - 3.3.1. MSI-X..... 37
  - 3.3.2. ATS..... 37
  - 3.3.3. Direct P2P ..... 37
  - 3.3.4. PASID ..... 37
  - 3.3.5. LNR..... 38
  - 3.3.6. TPH ..... 38
- 4 Device Security Architecture ..... 39
  - 4.1. Resource Isolation and Protection ..... 39
  - 4.2. Address Translation..... 39
  - 4.3. Device Resource ..... 39
  - 4.4. Device Identity and Measurement Reporting..... 39
    - 4.4.1. Device Firmware Resilience ..... 40
    - 4.4.2. Runtime Firmware Update ..... 40
  - 4.5. Secure Interconnects..... 42
  - 4.6. Device Attached Memory..... 42
  - 4.7. TDI Security ..... 42
  - 4.8. Data Integrity Errors..... 43
  - 4.9. Debug Modes..... 43
    - 4.9.1. Device Debug Interface ..... 43
  - 4.10. Device Reset ..... 44
    - 4.10.1. Conventional Reset ..... 44
    - 4.10.2. Function Level Reset (FLR) ..... 44
  - 4.11. Timing ..... 45
  - 4.12. Error Handling ..... 45
    - 4.12.1. Error Trigger ..... 45
    - 4.12.2. Error Notification..... 48
    - 4.12.3. Error Recovery..... 50
- 5 Summary ..... 52
- Appendix A: Intel® TDX Connect Interoperability ..... 53
  - A.1. TDX Connect Software Interoperability ..... 53

## Table of Contents

A.1.1. DOE.....	53
A.1.2. SPDM .....	54
A.1.3. IDE_KM.....	55
A.1.4. TDISP .....	56
A.2. TDX Connect Hardware Interoperability.....	57
A.2.1. IDE Stream.....	57
A.2.2. TDISP .....	58
<b>Appendix B: Secure Device Interface Lifecycle Example .....</b>	<b>66</b>
B.1. SPDM Management.....	66
B.1.1. SPDM Session Setup.....	66
B.1.2. SPDM Session Termination.....	66
B.1.3. SPDM Session Heartbeat.....	67
B.1.4. SPDM Session Key Update .....	67
B.1.5. Device Information Recollection .....	67
B.2. IDE Stream Management.....	68
B.2.1. IDE Stream Setup.....	68
B.2.2. IDE Stream Stop .....	69
B.2.3. IDE Stream Key Refresh .....	70
B.3. TDI Lifecycle Management.....	71
B.3.1. TDI Assignment .....	71
B.3.2. TDI Teardown.....	73
<b>References.....</b>	<b>74</b>
Standards.....	74
Web Resources.....	76

## Introduction and Scope

In the computer industry, hardware-based trusted execution environments (TEEs) are used to provide the confidential computing environment. In this document, such TEEs are referred to as Trusted Execution Environment VMs (TVMs) to distinguish them from traditional virtual machines (VMs). Today, multiple CPU vendors such as Intel, AMD, ARM, and RISC-V already published the solution to address the need based upon the new capability in the host CPU.

In order to achieve high performance in the data center, the host CPU may offload some workload to the device, such as Hardware Security Module (HSM) for crypto accelerating, Graphic Processing Unit (GPU) for AI processing, and Smart Network Interface Card (NIC) for network processing. In this case, the confidential computing environment is extended from the TVM to a portion of a device (TEE Device Interface, also known as TDI).

The TDI could be a Physical function (PF), a Virtual function (VF), or an Assignable Device Interface (ADI). To support such functions, the industry standard groups, such as PCI-SIG and Compute Express Link (CXL) Consortium, defined the new standard to support the use case, including Integrity and Data Encryption (IDE) and TEE Device Interface Security Protocol (TDISP).

In a host environment, we use TEE security manager (TSM) to indicate the logic Trust Computing Base (TCB) component to enforce the security policy. The TSM could be inside of a TVM, or the TSM could be a component outside of the TVM and trusted by the TVM.

Inside the device, we use Device Security Manager (DSM) to indicate the logical TCB component that enforces the security policy. The TSM should set up a secure management channel with the DSM to get the device information and manage the device interface. The TSM and DSM may also negotiate the encryption key for secure communication.

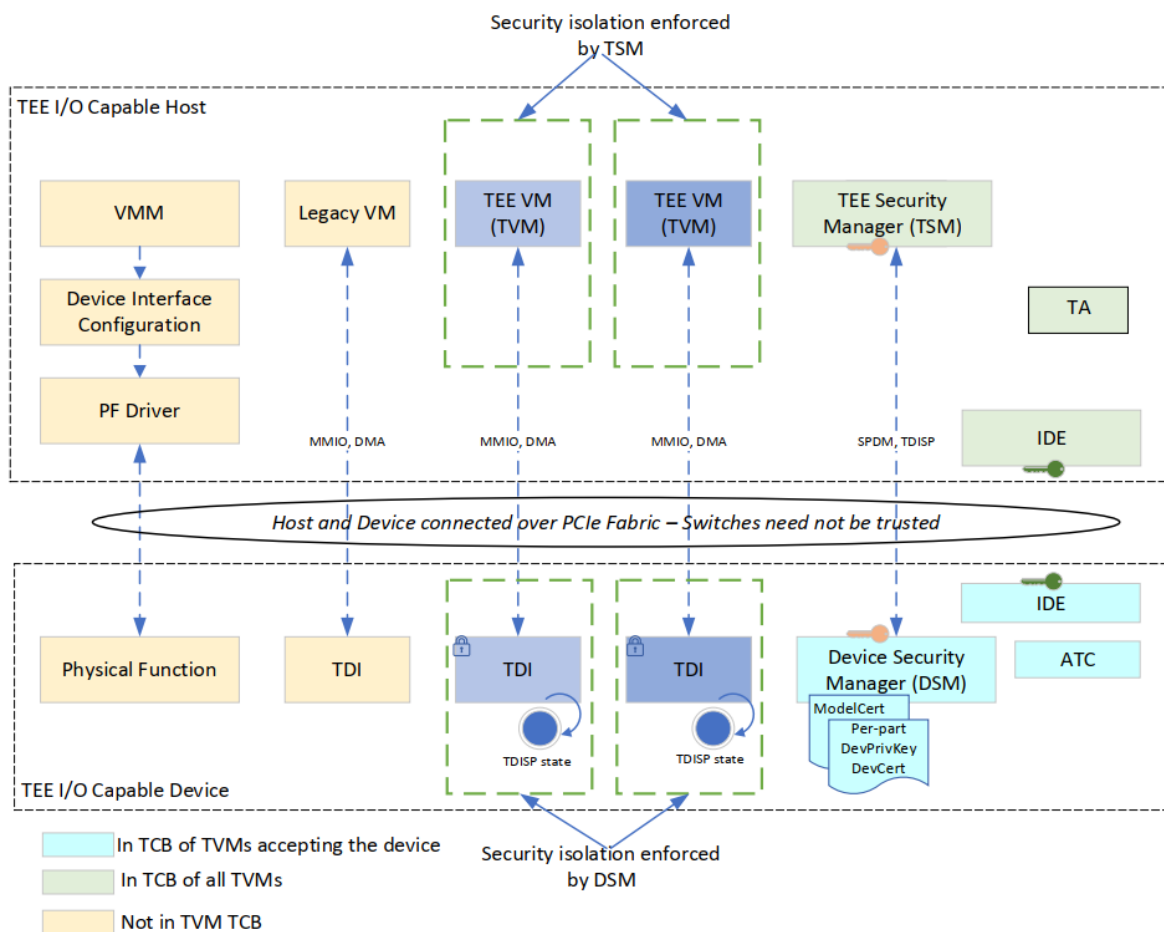


Figure 0-1: TEE-IO Component Conceptual View (Source: [PCIe TDISP 1.0])

Figure 0-1 shows a conceptual view of TEE-IO component as an example.

This whitepaper's goal is to provide the guidance on how to build a device to meet the confidential computing requirement, as such the TVM can offload the workload to the device.

The organization of the white paper is as follows:

- Chapter 1 provides an overview on how the host software establishes the trust relationship with the device.
- Chapter 2 describes the software protocols and requirements for secure management of SPDM and IDE session management between TSM and DSM and for secure management of TDI assignment to a TVM.
- Chapter 3 describes the device and the host hardware shared responsibilities for ensuring access controls of data path MMIO and DMA access controls between a TDI and its corresponding TVM.

- Chapter 4 describes intra-device security requirements including isolation, integrity, and confidentiality protection of TDI and TVM private data, device identity, and measurement reporting and device firmware update.
- Appendix A specifies the specific requirements for TEE-IO device interoperability with the TDX Connect architecture.

## Glossary

Table 0-1: Glossary

Term	Description
ACS	Access Control Service
ATS	Address Translation Service
CA	Completer Abort
CMA	Component Measurement and Authentication
CPL	Completion
DOE	Data Object Exchange
DSM	Device Security Manager
EP	Endpoint
FLR	Function Level Reset
IDE	Integrity and Data Encryption
IDE_KM	IDE Key Management
LN	Lightweight Notification
LNR	LN Request
MSI	Message Signal Interrupt
NPR	Non-Posted Request
P2P	Peer to Peer
PASID	Process Address Space ID
PR	Posted Request
PRS	Page Request Service

<b>Term</b>	<b>Description</b>
<b>RP</b>	Root Port
<b>SPDM</b>	Security Protocol and Data Model
<b>TDI</b>	TEE Device Interface
<b>TDISP</b>	TEE Device Interface Security Protocol
<b>TEE</b>	Trusted Execution Environment
<b>TLP</b>	Transaction Layer Packet
<b>TPH</b>	TLP Processing Hint
<b>TSM</b>	TEE Security Manager
<b>TVM</b>	TEE Virtual Machine
<b>UC</b>	Unexpected Completion
<b>UR</b>	Unsupported Request

# 1 TEE-IO Device Architecture Overview

This chapter provides an overview of the TEE-IO device architecture.

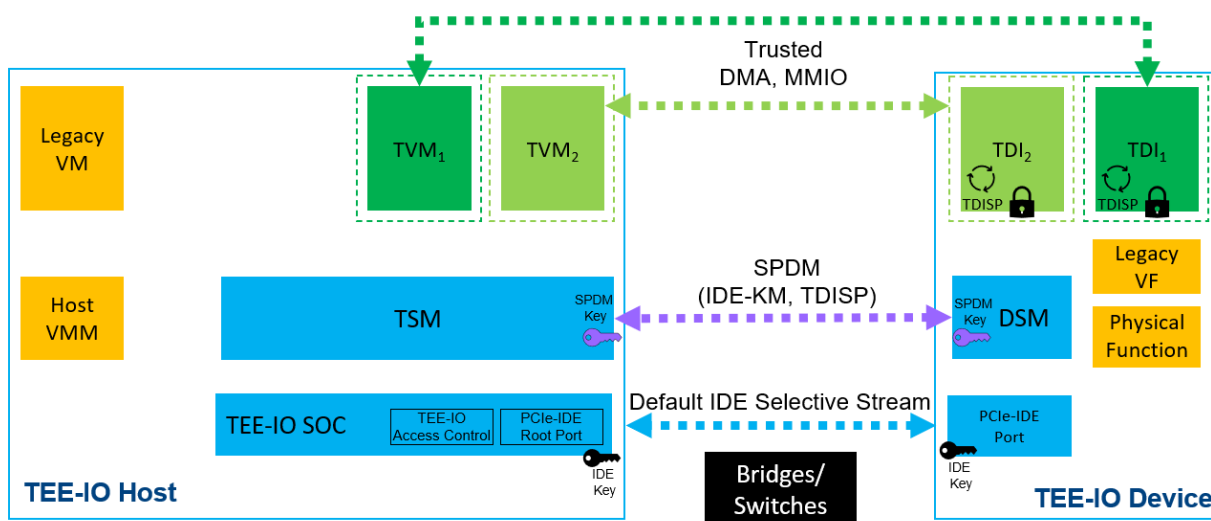
## 1.1. TEE-IO Security Model

According to [PCIe TDISP 1.0], the TEE-I/O security model is primarily intended to apply to systems using device resources directly assigned to VMs. The device resource can be assigned as a TEE-IO Device Interface (TDI). Before the TVM accepts a TDI, only the TSM and the host CPU are in the TEE TCB of the TVM. Once the TVM accepts a TDI, the TVM extends its TEE TCB to the entire TEE-IO device DSM, even if only one TDI from the device is accepted by the TVM.

Table 1-1 provides a summary of the shared security responsibility model of TDISP:

**Table 1-1: Component in TEE-IO Security Model**

Component	Role	Responsibilities
TSM	TEE TCB for TVM, a logic entity in a host.	Enforce security policies on the host.
DSM	TEE TCB for TVM, a logic entity in a device	Enforce security policies on the device.
TVM	TEE virtual machine on host	Admit the DSM into the TEE TCB. Accept the TDI.
TDI	Portion of the device assigned to a TVM	Provide device functions to a TVM.
VMM	Resource Manager	Attach and detach TDIs to a TVM.



**Figure 1-1: Component in TEE-IO**

## 1.2. Security Model for the Device

According to [PCIe TDISP 1.0] (11.1 Overview of the TEE-I/O Security Model as it Relates to Devices, page 11), the security objective for a TEE-IO device is to protect the TVM data, code, and execution state with respect to:

- **Confidentiality:** protection from disclosure to entities such as firmware, software, or hardware not in the TEE TCB of the TVM (e.g., other TVMs, VMM, etc.).
- **Integrity:** protection from modification by entities such as firmware, software, or hardware not in the TEE TCB of the TVM (e.g., other TVMs, VMM, etc.). Replay-protection of the TVM data is also in scope.

There is no requirement for the protection of TVMs against denial-of-service attacks.

To achieve these security objectives, the device shall support:

1. **Device Identity Authentication and Measurement Reporting.** In order to protect a TVM from TEE-IO device identity spoofing, the device shall implement a root-of-trust (ROT) for measurement (RTM), storage (RTS) and reporting (RTR) to support identity authentication and measurement reporting. The device debug interface shall not impact the device security properties. See Chapter 4 for detail.
2. **Authenticated Secure Communication.** The device shall use a secure communication channel to transfer the trusted data between the host and the device. The secure channel shall provide confidentiality, integrity, replay protection and message ordering protection. See Chapter 2 Secure Protocol and Data Model (SPDM) for management communication channel and Chapter 3 Integrity and Data Encryption (IDE) stream for data communication channel.
3. **TEE Device Interface (TDI) Management.** The device shall support locking down configurations of the TDI, reporting the configuration in a trusted manner, securely placing TDIs to operational state, and tearing them down securely when the TDI is detached. See Chapter 2 TEE Device Interface Security Protocol (TDISP) and Chapter 3 PCI Express Transaction Layer Packet (TLP) Rule for TDI.
4. **Device Security Architecture.** The device shall provide isolation and access control for the TVM data in the device for protection against the entities not in the trust boundary of TVM (such as the VMM, other TVMs, untrusted device components, untrusted physical functions, other TDIs). The device should implement Advanced Error Reporting (AER) to report errors according to [PCIe TDISP 1.0] page 13. See Chapter 4 for detail.

## 1.3. Implementing TEE-IO Device Stack

According to [PCIe TDISP 1.0] page 13, the device memory could be system level memory (defined as non-TEE memory), or TDI specific memory (defined as TEE memory). The TEE

memory shall have mechanisms to ensure the confidentiality and optional integrity of TVM data.

The device implementation to support TEE-IO can be separated as software stack and hardware stack. Usually, the **TEE-IO software stack** is to transfer management information, such as device identity, device measurement, data encryption key, TDI lock, TDI report, TDI attachment, TDI detachment, etc.

The **TEE-IO hardware stack** provides link encryption to ensure the link's confidentiality and integrity, downstream access control logic to prevent non-trusted MMIO access to TEE memory, and upstream logic to ensure DMA TEE data sent with encrypted IDE TLP.

## 1.4. Secure Device Interface Lifecycle Example

The following figures show the high-level software flow for TEE-IO architecture as a typical example.

Step 1 (SPDM setup). See Figure 1-2. The VMM calls the TSM to establish the SPDM session with the device. Then the TSM will own the SPDM session. The TSM will also collect the device certificate and measurement and return to VMM for device attestation later. This step is per device.

Step 2 (IDE Stream setup). See Figure 1-3. The VMM calls the TSM to use IDE\_KM to set up the default IDE selective stream with the device. The IDE\_KM is an SPDM vendor defined protocol sent in the SPDM session. The TSM will also configure the SOC IDE key programming register to ensure the IDE selective stream between SOC and device is established. After this step, the device and host SOC have two secure communication sessions. The SPDM session is a software session, which is used for software configuration such as IDE\_KM and TDISP. The IDE stream is hardware session, which is used to secure the PCI Express TLP. This step is also per device.

Step 3 (TDI assignment). See Figure 1-4. The VMM locks the TDI and launches a TVM, then assigns the TDI to the TVM.

- The TVM will get the device certificate and measurement from the TSM, then the TVM will verify the device based on a TVM specific policy. For example, the device certificate must have a trusted root Certificate Authority (CA) and the device measurement must match the latest reference manifest published by the device vendor. If the verification passes, the TVM can accept the TDI and use TDISP protocol to start the TDI. If the verification fails, the TVM will reject the device.
- After the device is approved by the TVM, the VMM sets up the DMA and MMIO for the TDI and lets the TVM accept the configuration.
- Now TVM can use trusted DMA and MMIO to communicate with the TDI. This step is per device interface.

The VMM can repeat the same process to assign another TDI to another TVM. See Figure 1-5. Note: One TVM may accept multiple TDIs. But one TDI must not be assigned to more than one TVM.

If a TDI is no longer needed, then the TVM or VMM can stop the TDI. If all TDIs in a device are removed, the VMM can remove the IDE stream and terminate the SPDM session for the device.

Appendix B provides more details on SPDM management, IDE Stream management, and TDI lifecycle management.

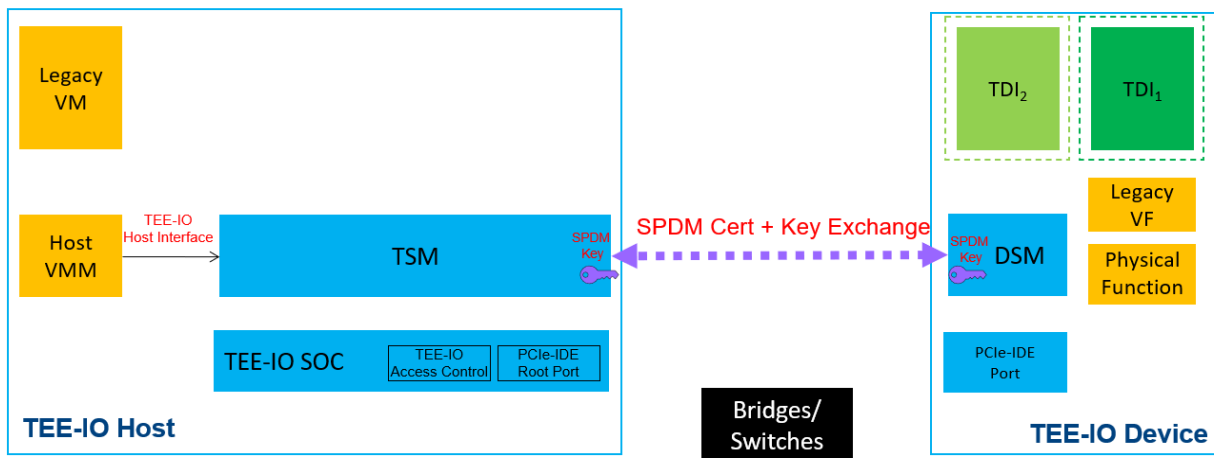


Figure 1-2: Device SPDM Session Setup for Secure Communication

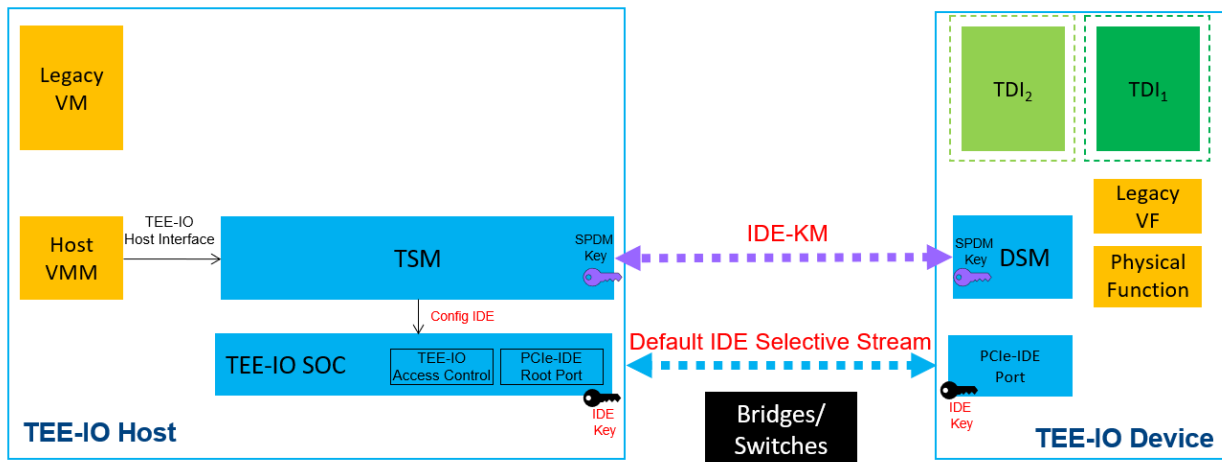


Figure 1-3: Device IDE Setup for Link Encryption

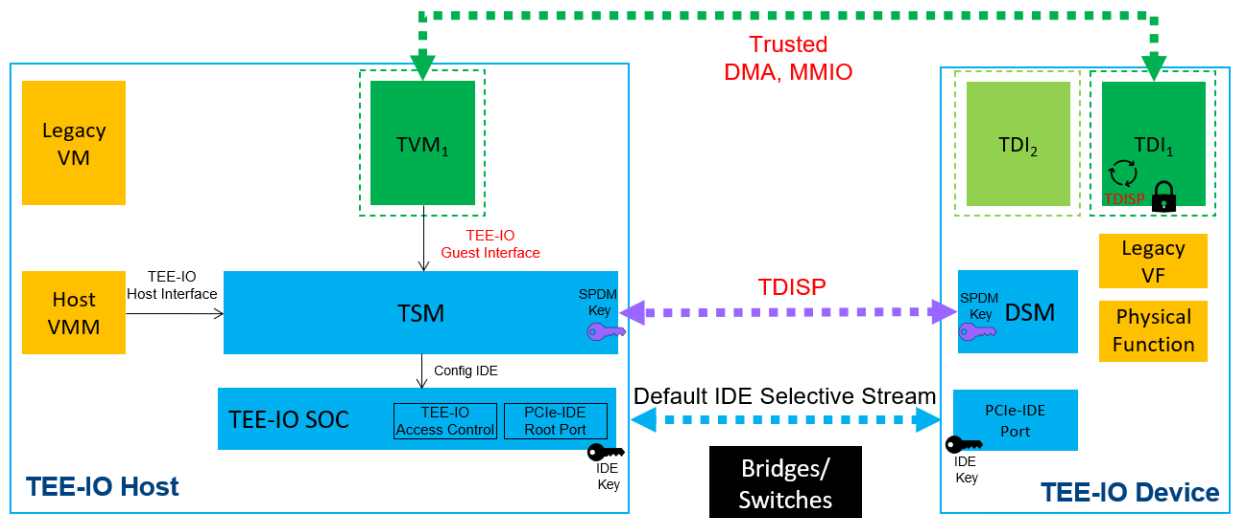


Figure 1-4: TVM Launch and TDI Start

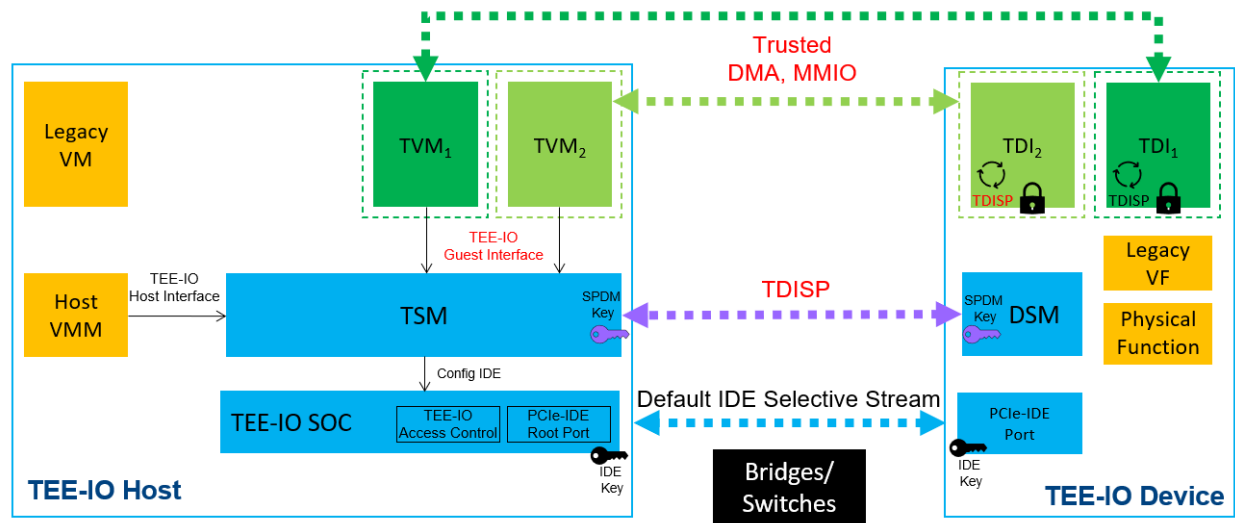


Figure 1-5: Another TVM Launch and TDI Start

## 2 TEE-IO Software Stack

This chapter describes the TEE-IO software stack requirement for the device. The device TEE-IO software stack implements the responder role of the secure device management protocols (i.e., SPDM, IDE\_KM and TDISP).

### 2.1. Software Stack Overview

Figure 2-1 shows one example of implementing the TEE-IO software stack in the device.

The purpose of the TEE Device Interface Security Protocol (TDISP) protocol is to manage the TDI. The purpose of the Integrity and Data Encryption Key Management (IDE\_KM) protocol is to configure the IDE keys for the PCIe device endpoint. The PCIe Root Port IDE Key can be configured via IDE\_KM or vendor specific mechanism. Both IDE\_KM and TDISP are application-level protocols that are transported within a Secure Protocol and Data Model (SPDM) secure session for transport level protection. The SPDM messages are sent and received via PCI Data Object Exchange (DOE) interface.

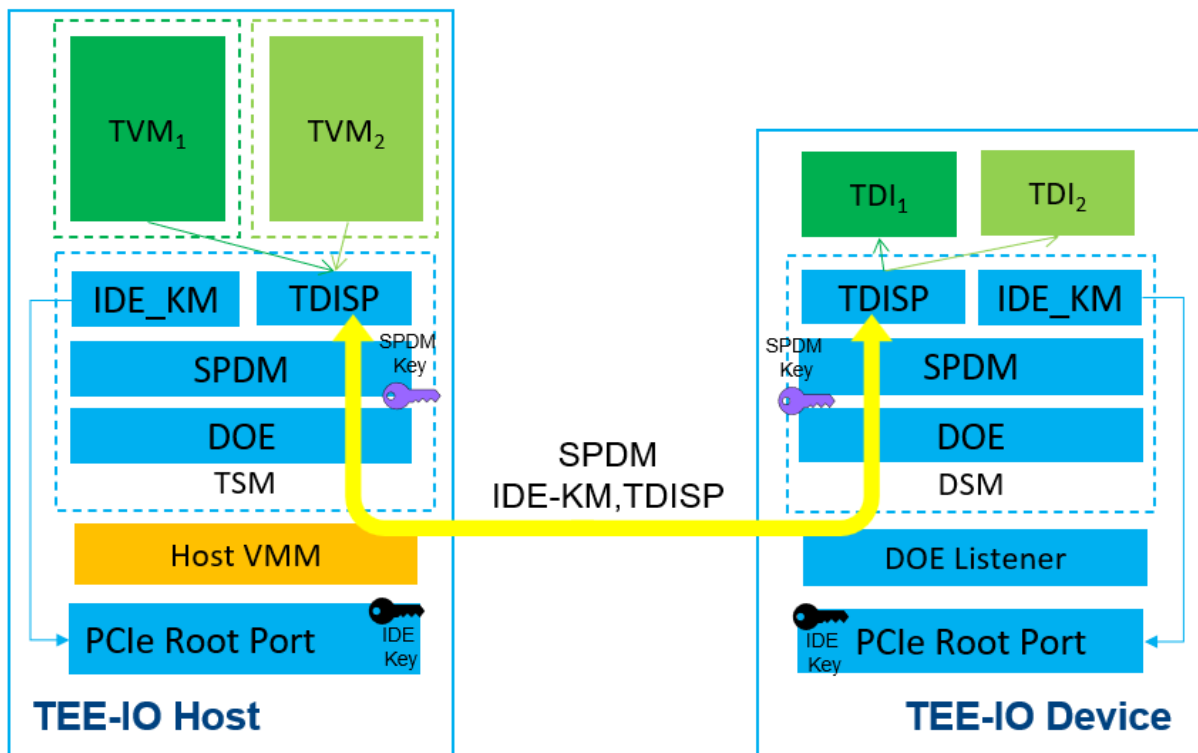


Figure 2-1: Device DSM Software Stack

On the host side, the VMM manages the DOE mailbox to send or receive SPDM messages. The TSM acts as the security policy enforce to encrypt and decrypt SPDM secure messages, including IDE\_KM message and TDISP message. At runtime, the VMM or TVM requests the

TSM to generate a protocol message request. Then the VMM sends the request to the device DOE mailbox. Later, the VMM receives the response from the DOE mailbox and sends back to TSM to process it.

On the device side, there should be a listener to wait for SPDM messages. Then the SPDM stack in DSM will decrypt the SPDM secure messages and dispatch to TDISP or IDE\_KM callback to process the TDISP or IDE\_KM request message. Later, DSM will encrypt the TDISP or IDE\_KM response message and sends to DOE mailbox on the device.

## 2.2. Device Communication (DOE)

PCI Express **Data Object Exchange (DOE)** ECN defines a mailbox mechanism for the host software to perform data object exchange with the device, such as a SPDM message or a secure SPDM message. The host uses DOE to exchange messages, such as IDE\_KM or TDISP, with the device. One device may support multiple DOE mailboxes.

### Requirement:

2.2.1. The device shall support SPDM over PCI DOE.

2.2.2. The device shall expose DOE Extended Capability registers for capability detection and control. [PCIe DOE 1.0] 7.9.xx Data Object Exchange (DOE) Extended Capability.

2.2.3. The device shall support the DOE 1.0 protocol including DOE Discovery (Data Object 0), CMA/SPDM (Data Object 1), Secure CMA/SPDM (Data Object 2). [PCIe DOE 1.0] 6.xx Data Object Exchange (DOE) and [PCIe IDE 1.0] 6.xx Data Object Exchange (DOE).

2.2.4. The device shall provide the DOE Extended Capability within function 0 to support the establishment of the SPDM session and transport secured messages. [PCIe TDISP 1.0] 11.2.2 TDISP message transport.

For TDX Connect compatibility, please refer to Appendix A “DOE” section.

## 2.3. Device Attestation (SPDM)

The TVM needs to offload the confidential workload to the device TDI. As such, the TVM needs to verify the device.

PCI Express **Component Measurement and Authentication (CMA)** ECN 1.0 defines the mechanism based upon Secure Protocol and Data Model (SPDM). The DSM provides the device certificate, authentication, and measurement reporting via SPDM.

### Requirement:

- 2.3.1. The device shall support SPDM version 1.2.
- 2.3.2. The device shall support SPDM capability: CERT\_CAP and MEAS\_CAP, and support GET\_DIGEST, GET\_CERTIFICATE, and GET\_MEASUREMENTS.
- 2.3.3. The device shall support at least one of BaseAsymAlgo listed in CMA ECN: TPM\_ALG\_RSASSA\_3072, TPM\_ALG\_ECDSA\_ECC\_NIST\_P256, TPM\_ALG\_ECDSA\_ECC\_NIST\_P384.
- 2.3.4. The device shall support at least one of BaseHashAlgo listed in CMA ECN: TPM\_ALG\_SHA\_256, TPM\_ALG\_SHA\_384.
- 2.3.5. The device shall support at least one of MeasurementHashAlgo listed in CMA ECN: TPM\_ALG\_SHA\_256, TPM\_ALG\_SHA\_384.
- 2.3.6. The device shall support MeasurementSpecification: DMTF (bit 0).
- 2.3.7. The device certificate shall follow DSP0274 requirement. The X.509 certificate shall follow SPDM 1.2.1 Table 33 – Required fields. The X.509 certificate OID shall follow SPDM 1.2.1, 10.8.2, SPDM certificate requirements and recommendations, including 10.8.2.1 Extended Key Usage authentication OIDs, 10.8.2.2 SPDM Non-Critical Certificate Extension OID.
- 2.3.8. The device shall only return the DMTFSpecMeasurementValueType defined in SPDM 1.2.1. Table 45 – DMTF measurement specification format.
- 2.3.9. The DICE device shall support ALIAS\_CERT\_CAP and return DICE alias certificate.
- 2.3.10. The DICE alias certificate shall include DiceTcbInfo OID. The DiceTcbInfo shall include firmware information such as digest and/or secure version number (SVN).

For TDX Connect compatibility, please refer to Appendix A “SPDM” section.

## 2.4. Software Secure Communication (SPDM)

The TSM needs to establish an authenticated secure session with the device for integrity and confidentiality. This software secure session is used to exchange the hardware encryption key such as IDE\_KM protocol, or TDI management such as TDISP protocol.

PCI Express **Component Measurement and Authentication (CMA)** ECN 1.0 Revised defines the mechanism based on **Secure Protocol and Data Model (SPDM)** for authenticated secure session between the host TSM and the device DSM. PCI Express Integrity and Data Encryption (IDE) ECN and TEE Device Interface Security Protocol (TDISP) ECN rely on SPDM for the secure management data communication.

### Requirement:

- 2.4.1. The device shall support SPDM capability: ENCRYPT\_CAP, MAC\_CAP, and KEY\_EX\_CAP, and support KEY\_EXCHANGE, FINISH, and END\_SESSION.
- 2.4.2. The device shall support at least one of DHE Group: secp256r1, secp384r1.
- 2.4.3. The device shall support AEAD Cipher Suite: AES-256-GCM.
- 2.4.4. The device shall support Key Schedule Algorithm: DMTF.
- 2.4.5. The device shall support DSP0274 OtherParamsSupport.OpaqueDataFmt1.
- 2.4.6. The device shall use DSP0277 version 1.1 as Secured Message transport binding version in the Secure Message opaque data.
- 2.4.7. The device should support KEY\_UPD\_CAP. If the device supports KEY\_UPD\_CAP, the device shall support update keys for both directions with UpdateAllKeys.
- 2.4.8. The device may support HBEAT\_CAP. If it is supported, the device session shall be terminated after twice HeartbeatPeriod.
- 2.4.9. The DSM shall teardown the session if a firmware update is triggered, if the SPDM 1.2 SessionPolicy.TerminationPolicy = 0.
- 2.4.10. The DSM shall keep the confidentiality of the SPDM session key.
- 2.4.11. The DSM may perform mutual authentication based upon the TSM capability during SPDM session establishment, or DSM may perform TVM authentication in an established SPDM session. This is use case specific.

**NOTE:** The TSM may be implemented in software, making it hard to provision a public certificate and a private key to sign the SPDM message at session creation. The TSM may use a non-standardized TEE-Quote based certificate as described in Remote-Attestation TLS (RA-TLS) to support quote-based mutual attestation. Or the TSM/DSM may use two-phase authentications: In phase 1 TSM attests the device as described in SPDM specification; in phase 2 DSM authenticates the host environment including host TEE TCB (CPU and TSM), host TVM, and so on with an implementation specific mechanism.

For TDX Connect compatibility, please refer to Appendix A “SPDM” section.

## 2.5. Link Encryption Key Management (IDE\_KM)

The TSM needs to setup link encryption with the device to mitigate possible attacks in the path between the host and device.

PCI Express **Integrity and Data Encryption (IDE)** ECN provides confidentiality, integrity, and replay protection for Transaction Layer Packet (TLP) transmitted and received between two

PCI Express ports. The TSM uses IDE key management (IDE\_KM) protocol to manage the IDE keys with the DSM, such as programming the IDE key, starting or stopping the IDE stream. The DSM configures the IDE encryption keys in the device. Please refer to Chapter 3, IDE Stream section, for an example on how to setup IDE stream, stop IDE stream, etc.

**Requirement:**

2.5.1. The DSM shall support IDE\_KM payload in SPDM vendor defined message. [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE\_KM), page 19.

2.5.2. The device shall expose DOE for IDE in Function 0. [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE\_KM), page 20.

2.5.3. The DSM shall support IDE\_KM messages QUERY. [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE\_KM), page 21.

2.5.4. The DSM shall support IDE\_KM messages KEY\_PROG. [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE\_KM), page 22.

2.5.5. The DSM shall support IDE\_KM messages SET\_K\_GO. [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE\_KM), page 23.

2.5.6. The DSM shall support IDE\_KM messages SET\_K\_STOP. [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE\_KM), page 24.

2.5.7. The DSM shall handle IDE enable/disable in IDE Stream Enable bit, according to [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE\_KM), page 24.

2.5.8. The DSM shall manage the Secure/Insecure State correctly in SPDM secure session, according to [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE\_KM), page 24.

2.5.9. The DSM shall handle key correctly, according to [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE\_KM), page 26. The key set bit shall be used for the key update.

For TDX Connect compatibility, please refer to Appendix A “IDE\_KM” section.

## 2.6. Device Interface Management (TDISP)

One device may be used by multiple TVMs. The VMM needs to assign the TDI in the device to the TVM. Also, the TVM needs to decide to start or stop the TDI.

PCI Express **TEE Device Interface Security Protocol (TDISP)** ECN defines such mechanism. The TVM and VMM may use TDISP protocol to manage the device TDI. The DSM provides the interface management, such as TDI configuration lock with policy (such as NO\_FW\_UPDATE) and reporting, and TDI attach and detach.

**Requirement:**

2.6.1. The DSM shall ensure the IDE\_KM (for IDE keys establishment) and TDISP (for TDI management) using same SPDM session. [PCIe TDISP 1.0] 11.1 Overview of the TEE-I/O Security Model as it Relates to Devices, page 15.

2.6.2. If implemented, the DSM shall follow the peer-to-peer (P2P) communication rule to set up P2P stream, according to [PCIe TDISP 1.0] 11.2 TDISP Rules, page 19.

2.6.3. The DSM shall support TDISP payload in SPDM vendor defined message, according to [PCIe TDISP 1.0] 11.2.2 TDISP Message Transport, page 21.

2.6.4. The DSM shall follow Requirements for Responders (DSM), according to [PCIe TDISP 1.0] 11.2.4 Requirements for Responders (DSM), page 22.

2.6.5. The DSM shall follow DSM Tracking and Handling of Locked TDI Configurations, according to [PCIe TDISP 1.0] 11.2.6 DSM Tracking and Handling of Locked TDI Configurations, page 23.

2.6.6. The DSM shall support TDISP required messages: TDISP\_VERSION, TDISP\_CAPABILITIES, LOCK\_INTERFACE\_RESPONSE, DEVICE\_INTERFACE\_REPORT, DEVICE\_INTERFACE\_STATE, START\_INTERFACE\_RESPONSE, STOP\_INTERFACE\_RESPONSE.

2.6.7. The DSM shall follow TDISP Message Format and Protocol Versioning, according to [PCIe TDISP 1.0] 11.3.3 TDISP Message Format and Protocol Versioning, page 28.

2.6.8. The DSM shall support GET\_TDISP\_VERSION according to [PCIe TDISP 1.0] 11.3.4 GET\_TDISP\_VERSION, page 30 and 11.3.5 TDISP\_VERSION, page 30.

2.6.9. The DSM shall support GET\_TDISP\_CAPABILITIES according to [PCIe TDISP 1.0] 11.3.6 GET\_TDISP\_CAPABILITIES, page 30 and 11.3.7 TDISP\_CAPABILITIES, page 31.

The DSM shall support a DEV\_ADDR\_WIDTH that is equal to or larger than host address width, to prevent address alias attack.

2.6.10. The DSM shall support LOCK\_INTERFACE\_REQUEST according to [PCIe TDISP 1.0] 11.3.8 LOCK\_INTERFACE\_REQUEST, page 31 and 11.3.9 LOCK\_INTERFACE\_RESPONSE, page 34.

2.6.11. The DSM shall support GET\_DEVICE\_INTERFACE\_REPORT according to [PCIe TDISP 1.0] 11.3.10 GET\_DEVICE\_INTERFACE\_REPORT, page 35 and 11.3.11 DEVICE\_INTERFACE\_REPORT, page 36. The DSM shall generate the TDI report according to [PCIe TDISP 1.0] Table 15 TDI Report Structure, page 37.

2.6.12. The DSM shall support GET\_DEVICE\_INTERFACE\_STATE according to [PCIe TDISP 1.0] 11.3.12 GET\_DEVICE\_INTERFACE\_STATE, page 40 and 11.3.13 DEVICE\_INTERFACE\_STATE, page 40.

2.6.13. The DSM shall support START\_INTERFACE\_REQUEST according to [PCIe TDISP 1.0] 11.3.14 LOCK\_INTERFACE\_REQUEST, page 40 and 11.3.15 START\_INTERFACE\_RESPONSE, page 41.

2.6.14. The DSM shall support STOP\_INTERFACE\_REQUEST according to [PCIe TDISP 1.0] 11.3.16 LOCK\_INTERFACE\_REQUEST, page 41 and 11.3.17 STOP\_INTERFACE\_RESPONSE, page 42.

2.6.15. If implemented, the DSM shall support BIND\_P2P\_STREAM\_REQUEST according to [PCIe TDISP 1.0] 11.3.18 BIND\_P2P\_STREAM\_REQUEST, page 42 and 11.3.19 BIND\_P2P\_STREAM\_RESPONSE, page 43.

2.6.16. If implemented, the DSM shall support UNBIND\_P2P\_STREAM\_REQUEST according to [PCIe TDISP 1.0] 11.3.20 UNBIND\_P2P\_STREAM\_REQUEST, page 43 and 11.3.21 UNBIND\_P2P\_STREAM\_RESPONSE, page 44.

2.6.17. If implemented, the DSM shall support SET\_MMIO\_ATTRIBUTE\_REQUEST according to [PCIe TDISP 1.0] 11.3.22 SET\_MMIO\_ATTRIBUTE\_REQUEST, page 44 and 11.3.23 SET\_MMIO\_ATTRIBUTE\_RESPONSE, page 45.

2.6.18. The DSM shall support TDISP\_ERROR according to [PCIe TDISP 1.0] 11.3.24 TDISP\_ERROR, page 46.

2.6.19. If implemented, the DSM shall support VDM\_REQUEST according to [PCIe TDISP 1.0] 11.3.25 VDM\_REQUEST, page 48 and 11.3.26 VDM\_RESPONSE, page 48.

For TDX Connect compatibility, please refer to Appendix A “TDISP” section.

## 2.7. Implementation Reference

### 2.7.1. SPDM software stack

The DMTF open sourced SPDM sample implementation at <https://github.com/DMTF/libspdm>.

**Table 2-1: SPDM software Stack Reference**

Component	Purpose	URL
spdm_responder_lib	Responder library. It can be used on the device side.	<a href="https://github.com/DMTF/libspdm/tree/main/library/spdm_responder_lib">https://github.com/DMTF/libspdm/tree/main/library/spdm_responder_lib</a>
spdm_requester_lib	Requester library. It can be used to test the device.	<a href="https://github.com/DMTF/libspdm/tree/main/library/spdm_requester_lib">https://github.com/DMTF/libspdm/tree/main/library/spdm_requester_lib</a>

spdm_device_secret_lib_sample	sample device library to support measurement reporting and digital signature generation on the device side	<a href="https://github.com/DMTF/libspdm/tree/main/os_stub/spdm_device_secret_lib_sample">https://github.com/DMTF/libspdm/tree/main/os_stub/spdm_device_secret_lib_sample</a>
spdm-emu	An SPDM requester emulator, which may be used to test the device SPDM stack.	<a href="https://github.com/DMTF/spdm-emu">https://github.com/DMTF/spdm-emu</a>
spdm-dump	A tool to dump SPDM messages with PCAP format.	<a href="https://github.com/DMTF/spdm-dump">https://github.com/DMTF/spdm-dump</a>
SPDM-Responder-Validator	A test suite for the SPDM device	<a href="https://github.com/DMTF/SPDM-Responder-Validator">https://github.com/DMTF/SPDM-Responder-Validator</a>

### 2.7.2. IDE\_KM software stack

The DMTF open sourced SPDM sample implementation includes an IDE\_KM software example.

Table 2-2: IDE\_KM software Stack Reference

Component	Purpose	URL
pci_ide_km_responder_lib	Responder library. It can be used on the device side.	<a href="https://github.com/DMTF/spdm-emu/tree/main/library/pci_ide_km_responder_lib">https://github.com/DMTF/spdm-emu/tree/main/library/pci_ide_km_responder_lib</a>
pci_ide_km_requester_lib	Requester library. It can be used to test the device.	<a href="https://github.com/DMTF/spdm-emu/tree/main/library/pci_ide_km_requester_lib">https://github.com/DMTF/spdm-emu/tree/main/library/pci_ide_km_requester_lib</a>
pci_ide_km_device_lib_sample	sample device library to support IDE_KM messages.	<a href="https://github.com/DMTF/spdm-emu/tree/main/library/pci_ide_km_device_lib_sample">https://github.com/DMTF/spdm-emu/tree/main/library/pci_ide_km_device_lib_sample</a>

### 2.7.3. TDISP software stack

The DMTF open sourced SPDM sample implementation includes a TDISP software example.

Table 2-3: TDISP software Stack Reference

Component	Purpose	URL
pci_tdisp_responder_lib	Responder library. It can be used on the device side.	<a href="https://github.com/DMTF/spdm-emu/tree/main/library/pci_tdisp_responder_lib">https://github.com/DMTF/spdm-emu/tree/main/library/pci_tdisp_responder_lib</a>

pci_tdisp_requester_lib	Requester library. It can be used to test the device.	<a href="https://github.com/DMTF/spdm-emu/tree/main/library/pci_tdisp_requester_lib">https://github.com/DMTF/spdm-emu/tree/main/library/pci_tdisp_requester_lib</a>
pci_tdisp_device_lib_sample	sample device library to support IDE_KM messages.	<a href="https://github.com/DMTF/spdm-emu/tree/main/library/pci_tdisp_device_lib_sample">https://github.com/DMTF/spdm-emu/tree/main/library/pci_tdisp_device_lib_sample</a>

## 3 TEE-IO Hardware Stack

This chapter describes the TEE-IO hardware stack requirement on the device side. The TEE-IO hardware stack is the data encryption engine to support trusted MMIO and trusted DMA, and hardware registers such as data encryption key, TDI state, etc.

### 3.1. IDE Stream

IDE secures the TLP traffic from one port to another port. IDE TLP Prefix includes a “**T bit**”, which indicates the TLP originated from within a trusted execution environment. The “T bit” provides a mechanism to distinguish TLPs that are associated with a TVM. IDE mechanisms ensure that the T bit (like other TLP content) is secured during transit.

IDE Stream State Machine includes Insecure or Secure state. See figure 3-1.

**Secure State:** The device IDE Stream shall enter Secure State only after all necessary steps are done, including the keys are programmed, and the IDE stream is enabled.

**Insecure State:** The device IDE Stream shall enter an Insecure state if any necessary steps are not met. For example, the IDE stream is disabled, or IDE Check Failed error happens.

**Ready sub-state:** This is a sub-state of Insecure state. The device has all keys programmed. The only rest step is to trigger IDE stream. The trigger action requires both K\_SET\_GO messages and IDE Stream Enable bit set in the IDE Extended Capability Register.

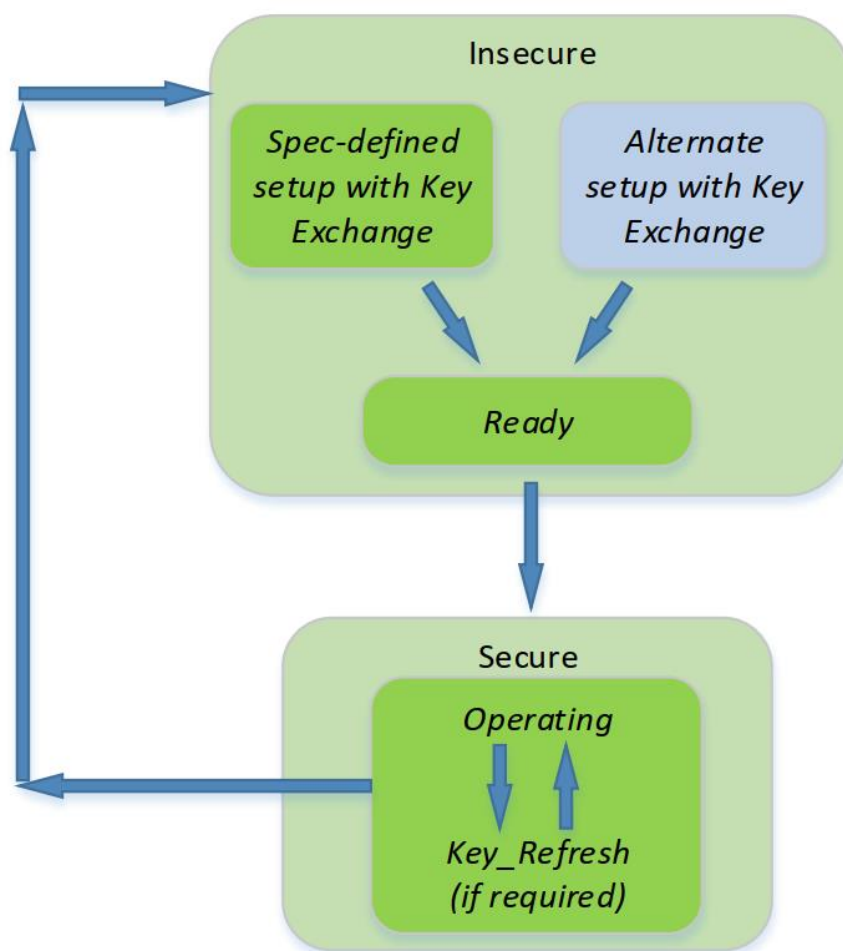


Figure 3-1: IDE Stream State Machine (Source: [PCIe IDE 1.0])

### Requirement:

3.1.1. The device shall maintain the IDE Stream Secure State and Insecure State, according to [PCIe IDE 1.0] 6.99.1 IDE Stream and TEE State Machines, page 16.

3.1.2. The device shall implement IDE Extended Capability in function 0. [PCIe IDE 1.0] 6.99.2 IDE Stream Establishment, page 18.

3.1.3. The device shall follow IDE TLP rule, according to [PCIe IDE 1.0] 6.99.4 IDE TLPs, page 29. The device shall use IDE TLP Prefix for all IDE TLPs.

3.1.4. The device shall follow a selective IDE stream rule, according to [PCIe IDE 1.0] 6.99.4 IDE TLPs, page 33. The device shall follow "Table XX – TLP Types for Selective IDE Streams" to only permit MRd, MRdLk, MWr, CfgRd1, CfgWr1, DMWr, Msg\*, MsgD\*, Cpl, CplD, CplLk, CplDLk, FetchAdd, Swap, and CAS. The IORd, IOWr, CfgRd0, and CfgWr0 are not encrypted.

### 3 TEE-IO Hardware Stack

- 3.1.5. The device shall follow IDE TLP sub-stream rule, according to [PCIe IDE 1.0] 6.99.5 IDE TLP Sub-streams, page 38.
- 3.1.6. The device shall follow IDE TLP Aggregation, if it is supported, according to [PCIe IDE 1.0] 6.99.6 IDE TLP Aggregation, page 40.
- 3.1.7. The device shall follow other IDE Rules for Non-Posted IDE Requests, according to [PCIe IDE 1.0] 6.99.8 Other IDE Rules, page 44.
- 3.1.8. The device shall follow other IDE Rules for resets, according to [PCIe IDE 1.0] 6.99.8 Other IDE Rules, page 44. Conventional Reset or Function Level Reset (FLR) to a function with IDE shall change to Insecure state. FLR to a function without IDE shall not affect IDE operation.
- 3.1.9. The device shall follow other IDE Rules for Access Control Services (ACS), according to [PCIe IDE 1.0] 6.99.8 Other IDE Rules, page 44.
- 3.1.10. The device shall follow other IDE Rules for error handling, according to [PCIe IDE 1.0] 6.99.8 Other IDE Rules, page 45. Detecting IDE Check Failed error, MAC check failure, underflow or overflow of TLP counter shall change to Insecure State.
- 3.1.11. The device shall follow other IDE Rules for power management, according to [PCIe IDE 1.0] 6.99.8 Other IDE Rules, page 46. No\_Soft\_Reset bit shall be Set. All state related to keys and counters shall be maintained in D0, D1, D2, and D3hot. They may be maintained in D3cold.
- 3.1.12. The device shall follow other IDE Rules for secure local environment, according to [PCIe IDE 1.0] 6.99.8 Other IDE Rules, page 46. Attempts to modify IDE registers, BARs, and other structures that could affect security or an IDE Stream shall be detected and enter Insecure state.
- 3.1.13. The device shall implement IDE Extended Capability, according to [PCIe IDE 1.0] 7.9.99 IDE Extended Capability, page 48.
- 3.1.14. The device shall expose "TEE-IO Supported" in Device Capability Register. [PCIe TDISP 1.0] 6.33.4 IDE TLPs, page 5.
- 3.1.15. The device shall implement bit 7 of Sub-Stream field as Reserved. [PCIe TDISP 1.0] 6.33.4 IDE TLPs, page 5.
- 3.1.16. The device shall follow rule when TEE-IO Supported is set, according to [PCIe TDISP 1.0] 6.33.4 IDE TLPs, page 6.
- 3.1.17. The device shall follow rule for Non-Posted IDE Requests based upon TEE-IO Supported bit, according to [PCIe TDISP 1.0] 6.33.8 Other IDE Rules, page 7.

### 3.1.1. IDE Stream Precedence

#### Requirement:

3.1.18. The device shall follow IDE stream precedence rule, according to [PCIe IDE 1.0] 6.99.4 IDE TLPs, page 37. The rule is summarized as follows:

a. For transmitter:

```

if (IAAR and IRAR rule hit) {
    Stream ID := corresponding Stream ID
} else if (default Selective IDE Stream is configured) {
    Stream ID := default Selective IDE Stream ID
} else if (Link IDE Stream is enabled) {
    Stream ID := Link IDE Stream ID
} else {
    Stream ID := invalid // Not use IDE-TLP
}

```

Where IAAR == IDE Address Association Register, IRAR = IDE RID Association Register.

b. For receiver:

```

Stream ID := Stream ID in the received IDE TLP Prefix

```

### 3.1.2. TEE Limited Stream

TEE limited stream is optional. It indicates that only those TLPs that have the T bit Set are permitted to be **associated with** this Stream. This is for optimization purposes, because not all data in TVM requires protection.

**Note:** This IDE feature requires the same capability to be supported and configured on the host Root Port to function correctly. For more details, please refer to [PCIe IDE 1.0].

#### Requirement if implemented:

3.1.19. If implemented, the device shall follow the description for TEE Limited Stream, according to [PCIe TDISP 1.0] 7.9.26.5.2: Selective IDE Stream Control Register, page 8.

### 3.1.3. Partial Header Encryption

Partial header encryption is added in [PCIe 6.0] as an optional feature. It provides the ability to reduce potential exposure to side-channel attacks by encrypting some portions of the Header

of an IDE Memory Request while maintaining information that is required for TLP routing and low-level TLP processing in the clear.

**Note:** This IDE feature requires the same capability to be supported and configured on the host Root Port to function correctly. For more details, please refer to [PCIe IDE 1.0].

**Requirement if implemented:**

3.1.20. If implemented, the device shall follow the description for partial header encryption, according to [PCIe 6.0] 7.9.26.2 IDE Capability Register, 7.9.26.4.1 Link IDE Stream Control Register, 7.9.26.5.2 Selective IDE Stream Control Register.

For TDX Connect compatibility, please refer to Appendix A “IDE Stream” section.

## 3.2. TDI TLP Rule

The device function unit may have:

- **TDI:** A Trusted Device Interface. It may be an entire device, a physical function (PF) or virtual function (VF). It may be in CONFIG\_UNLOCK, CONFIG\_LOCK, RUN, or ERROR state. A TDI shall be isolated from non-TDI and other TDIs by the DSM.
- **Non-TDI:** A legacy device function unit, which cannot be assigned as a TDI by definition. It may be a physical function (PF) or virtual function (VF).
- **DSM:** A device security manager, which is the TEE TCB for all TDIs.

A TDI state machine includes 4 states (see Figure 3-2):

**CONFIG\_UNLOCKED:** This is the initial, default state of a TDI. There is no security property (confidentiality and optional integrity) that a TDI needs to provide to the TVM data. The DSM shall not allow interface to start.

**CONFIG\_LOCKED:** This is the intermediate transition state from default to RUN. To enter this state, the VMM should finish configuration, and TSM should explicitly send LOCK\_INTERFACE. The DSM or TDI is expected to clean up and prepare to provide security property. For example, all previous MMIO requests or DMA requests should be dropped. From this point onwards, the device strictly enforces protection of all identified configuration parameters sensitive to the secure operation of the TDI. The device may also choose to implement checking of specific parameters where needed to confirm a safe secure configuration.

**RUN:** This is the actual functional state. TVM can send START\_INTERFACE to explicitly request the TDI to enter this state. The TDI can perform trusted MMIO or DMA transaction to communicate with TVM.

**ERROR:** This is the error state. When the DSM or TDI detects any change impacting device configuration or security in CONFIG\_LOCKED or RUN state, the TDI shall be changed to ERROR state. TDI shall not expose any confidential TVM data. TDI may start cleaning up the TVM data.

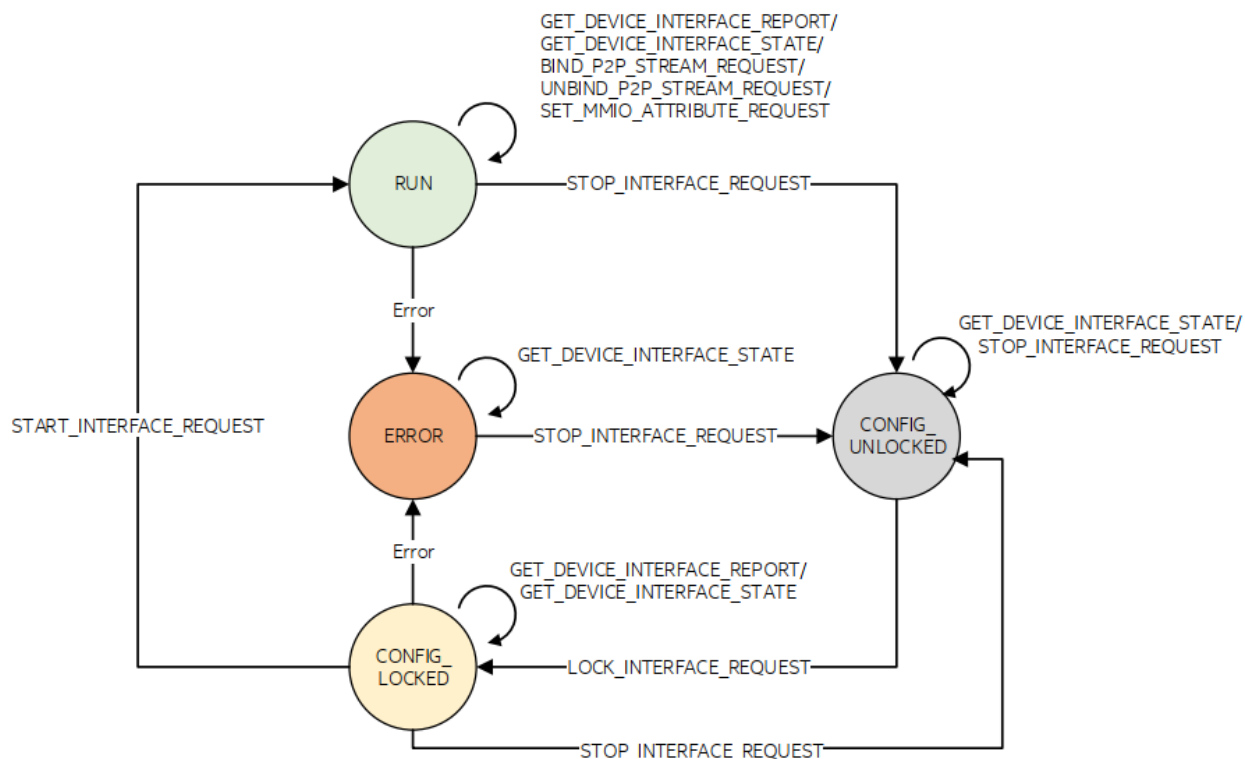


Figure 3-2: TDI State Machine (Source: [PCIe TDISP 1.0])

The host function unit may have

- **TEE VM (TVM):** A VM requiring TEE capability. A TVM shall be isolated from VMM, legacy VM, and other TVM by the TSM.
- **Legacy VM:** A VM not requiring TEE capability.
- **VMM:** A system resource manager for TVM and Legacy VM.
- **TSM:** A TEE security manager, which is TCB for all TVMs.

The connection between the device and the host:

- **TVM <-> TDI:** Follow TLP rules for MMIO, DMA, ATS, etc.
- **Legacy VM <-> TDI:** Not the focus of the TDISP specification. \*
- **Legacy VM/VMM <-> Non-TDI:** Legacy behavior. Out of scope.
- **TSM <-> DSM:** Follow SPDM, IDE\_KM, TDISP protocol.

\* NOTE: For Legacy VM <-> TDI, it is permitted for IDE streams established by the TSM to be used to carry TLPs associated with legacy VMs. [PCIe TDISP 1.0] 11.1 page 14. For a TDI that supports

assignment to Legacy VMs, if a TDI is assigned to a Legacy VM, the VMM assigns the TDI in CONFIG\_UNLOCKED, and the TSM must ensure that the TDI remains in that state unless and until the TDI is removed from the Legacy VM and prepared for re-assignment to a TDI. [PCIe TDISP 1.0] 11.1 page 18.

According to [PCIe TDISP 1.0], a device implementation may support legacy VM <> TDI. If a TDI is assigned to a legacy VM, the TDI is allowed to transmit or receive messages (such as MMIO or DMA) in Non-IDE Stream in CONFIG\_UNLOCK state. Once the TDI is asked to transit to CONFIG\_LOCK state, the TDI should drain all pending requests and receive data in CONFIG\_UNLOCK state.

In order to facilitate the TLP rule discussion, we define the following terms:

- **Bound IDE stream:** The IDE stream bound to a TDI. In normal case, it is the **default stream** in LOCK\_INTERFACE\_REQUEST [PCIe TDISP 1.0] 11.3.8. The IDE specification defined “default stream” in [PCIe IDE 1.0] 7.9.99.5.2 Selective IDE Stream Control Register. If a device includes multiple TDIs, those TDIs may share the same default stream. In direct peer to peer (P2P) case, the Bound IDE stream is the **P2P stream** in BIND\_P2P\_STREAM\_REQUEST [PCIe TDISP 1.0] 11.3.18.
- **Non-Bound IDE Stream:** The IDE stream not bound to this TDI.
- **Non-IDE Stream:** The plaintext TLP (not encrypted).
- **TEE-TLP:** TLP Bound IDE stream with T=1.
- **Non-TEE-TLP:** All the other not TEE-TLP, such as TLP Bound IDE stream with T=0, Non-Bound IDE Stream with T=0 or T=1, or Non-IDE Stream.

NOTE: The “T bit” indicates that the TLP originated **from** within a trusted execution environment (TEE). However, there is no bit to indicate if **the target** is within TEE or non-TEE. This is a known limitation so far. As such, the device should not use a common shared cache to store the data, unless the device cache has an attribute to identify if the cache-entry is private (TEE) or shared (non-TEE).

NOTE: the following rules are only for TDI, but not for non-TDI.

### 3.2.1. TDI as Completer

Resource definition:

- **TEE MMIO (T-MMIO):** The TEE memory in the device, read/write from the host, which must have mechanisms to ensure the confidentiality of TVM data, and optionally integrity. [PCIe TDISP 1.0] 11.1. page 13. It is NON\_TEE\_MEM=0 in the TDI report structure. TEE MMIO is only present when TDI is in CONFIG\_LOCK, RUN, or ERROR state. TEE MMIO does not exist when TDI is in CONFIG\_UNLOCK state.
- **Non-TEE MMIO (NT-MMIO):** The Non-TEE memory in the device, read/write from the host, which does not have above protection mechanism. It is NON\_TEE\_MEM=1 in the TDI report structure.

### 3 TEE-IO Hardware Stack

- **CFG:** The device configuration space read/write from the host. It is not required that Configuration Requests to a TDI be secured. [PCIe TDISP 1.0] 11.2. page 20.

Rule definition:

- **Success (V):** The device shall process and return success completion (SC) TLP for NPR.
- **Reject (X):** The device shall return an Unsupported Request (UR) TLP or drop the received TLP.

NOTE:

1. "If the result is a determination that the TLP must be rejected, **the associated TDI must transition to ERROR where indicated, but no further error reporting or logging is required to be performed on that TLP,**" [PCIe TDISP 1.0] 11.2. TDISP Rules Page 18. Here the "rejected" means the "IDE Check Failed" instead of access control rule. "no further error reporting is required" because there is no trusted way to guarantee that the device error message is delivered to the TVM. The man-in-the-middle adversary can block or delay the error reporting regardless of whether it is a hardware mechanism or a software mechanism. Please refer to Chapter 4, Error Handling Section.

2. "If the result is a determination that the **TLP must be rejected**, ... it is optionally permitted on a **case-by-case** basis to handle a Request as an **Unsupported Request**, and/or handle a Completion as an **Unexpected Completion**, or that the TLP be **dropped**." [PCIe TDISP 1.0] 11.2. TDISP Rules Page 18. In the case of rejection, the TLP can be responded with UR/UC or dropped.

Table 3-4, Table 3-5 and Table 3-6 are the rule summary of TDI acting as a Completer, according to [PCIe TDISP 1.0] 11.2.1 TLP Rules, page 21.

**Table 3-4: TEE-MMIO Rule Summary for TDI as Completer**

Access Control	CONFIG_UNLOCK	CONFIG_LOCK	RUN	ERROR
TEE-TLP	N/A *	T-MMIO (X)	T-MMIO (V)	T-MMIO (X)
Non-TEE-TLP	N/A **	T-MMIO (X)	T-MMIO (X)	T-MMIO (X)

\* By definition, the Bound IDE stream is only known after LOCK\_INTERFACE\_REQUEST is sent. As such, there is no TEE-TLP in CONFIG\_UNLOCK. All TLPs in CONFIG\_UNLOCK are non-TEE-TLP.

\*\* By definition, the TDI does not provide confidentiality or optional integrity of the TVM data in CONFIG\_UNLOCK, if the TDI is assigned to a legacy VM. As such, the TDI does not have any TEE-MMIO, because a TDI is not required to protect confidential data placed into it in this state. In this case, Non-TEE-TLP (no IDE or IDE with T=0) shall be accepted. TEE-TLP (IDE with T=1) from any TVM shall be rejected according to [PCIe TDISP 1.0] page 18.

**Table 3-5: NON-TEE-MMIO Rule Summary for TDI as Completer**

Access Control	CONFIG_UNLOCK	CONFIG_LOCK	RUN	ERROR
----------------	---------------	-------------	-----	-------

<b>TEE-TLP</b>	N/A	NT-MMIO (V)	NT-MMIO (V)	NT-MMIO (V)
<b>Non-TEE-TLP</b>	NT-MMIO (V)	NT-MMIO (V)	NT-MMIO (V)	NT-MMIO (V)

Table 3-6: CFG Rule Summary for TDI as Completer

Access Control	CONFIG_UNLOCK	CONFIG_LOCK	RUN	ERROR
<b>TEE-TLP</b>	N/A	CFG (V)	CFG (V)	CFG (V)
<b>Non-TEE-TLP</b>	CFG (V)	CFG (V)	CFG (V)	CFG (V)

NOTE:

1. **TEE MMIO:** TEE MMIO is only allowed with bound IDE stream, T=1, and TDI in RUN.

```

if ((TDI.STATE != RUN) ||
    (TLP.T == 0) ||
    (STREAM_TYPE != IDE_TLP) ||
    (IDE_STREAM_ID != TDISP_BOUND_STREAM_ID) ||
    (TLP.ADDRESS NOT IN STREAM-ASSOCIATION-RANGES[N])) {
    Access = Deny;
} else {
    Access = Allow;
}

```

2. **Non-TEE MMIO:** [PCIe TDISP 1.0] mentioned the rule for TDI acting as a Completer: “Requests targeting device memory received with the T bit Set while in any state other than RUN must be rejected.”. That rule is superseded by “The TDI’s handling is not modified by TDISP state for Received Memory Requests targeting MMIO with IS\_NON\_TEE\_MEM Set.”

3. **CPL rule:** The value of the T bit in the Completion(s) returned by the TDI **must match** the value of the T bit in the corresponding Request. [PCIe TDISP 1.0].

4. **CFG rule:** It is not required that Configuration Requests to a TDI be secured. [PCIe TDISP 1.0] 11.2. page 20. [PCIe IDE 1.0] Table XX–TLP Types for Selective IDE Streams mentions Type0 is not permitted for selective IDE streams, because we do not expect that will happen. A proper requester shall always use Type1. An improper requester may use Type0. The device may choose to accept or reject, but it does not impact security.

### 3.2.2. TDI as Requester

Resource definition:

- **DMA:** The host memory-read/write from the device. The device does not know if the host memory is TEE memory or non-TEE memory, according to [PCIe TDISP 1.0].
- **MSI:** The message signaled interrupt from the device to the host.
- **Trusted-MSI (T-MSI):** The LOCK\_MSIX flag in LOCK\_INTERFACE\_REQUEST is 1 and MSIX table is part of locked MMIO\_RANGE.

Rule definition:

- **Allowed (V):** The device can send the TLP.
- **Not allowed (X):** The device shall not send the TLP. It is DSM/TDI's responsibility.

Table 3-7, Table 3-8, and Table 3-9 are rule summary of TDI acting as a Requester, according to [PCIe TDISP 1.0] 11.2.1 TLP Rules, page 20.

**Table 3-7: DMA Rule Summary for TDI as Requester**

Operation	CONFIG_UNLOCK	CONFIG_LOCK	RUN	ERROR
TEE-TLP	N/A	DMA (X)	DMA (V)	DMA (X)
Non-TEE-TLP	DMA (V) **	DMA (X)	DMA (X)	DMA (X)

\*\* DMA is allowed in CONFIG\_UNLOCK, if the TDI is designed to a Legacy VM.

**Table 3-8: MSI Rule Summary for TDI as Requester**

Operation	CONFIG_UNLOCK	CONFIG_LOCK	RUN	ERROR
TEE-TLP	N/A	MSI (X)	MSI (X)	MSI (X)
Non-TEE-TLP	MSI (V)	MSI (V)	MSI (V)	MSI (V)

**Table 3-9: Trusted-MSI Rule Summary for TDI as Requester**

Operation	CONFIG_UNLOCK	CONFIG_LOCK	RUN	ERROR
TEE-TLP	N/A	T-MSI (X)	T-MSI (V)	T-MSI (X)
Non-TEE-TLP	N/A **	T-MSI (X)	T-MSI (X)	T-MSI (X)

\*\* Trusted-MSI is not applicable, only non-trusted MSI is allowed.

NOTE:

1. **DMA rule:** DMA is only allowed with bound IDE stream, T=1, and when TDI in RUN.

```
if (TDI.STATE != RUN) {
    Operation = Deny;
```

### 3 TEE-IO Hardware Stack

```

    } else {
        Operation = Allow;
        TLP.T = 1;
        STREAM_TYPE = IDE_TLP;
        IDE_STREAM_ID = TDISP_BOUND_STREAM_ID;
    }

```

2. **CPL** rule: For Memory Reads issued by the TDI while in RUN, the corresponding Completion(s) must be handled normally if and only if the TDI is still in RUN and must otherwise be rejected. [PCIe TDISP 1.0]. “Still in RUN” means that the TDI shall not change to ERROR state then back to RUN state again.

A TDI in RUN **must ignore** the value of the T bit in Received Completions. [PCIe TDISP 1.0]. The reason is that the host SOC implementation may set T=0 for non-TEE owned shared memory.

Receipt of a Completion with UR/CA or Completion timeout (following recovery retries) for request initiated by a TDI in CONFIG\_LOCK, RUN (with T=1) indicates occurrence of an uncorrectable error, TDI must transition to ERROR. [PCIe TDISP 1.0] 11.4.3. Securing Interconnects.

3. **MSI** rules: The T-bit must be set according to LOCK\_MSIX flag in LOCK\_INTERFACE\_REQUEST and MSIX table is part of locked MMIO\_RANGE, if TDI is in RUN state.

```

    if ((TDI.STATE == RUN) &&
        (LOCK_MSIX flag == 1) &&
        (MSIX table is part of locked MMIO_RANGE)) {
        TLP.T = 1;
    } else {
        TLP.T = 0;
    }

```

An MSI with T-bit clear is always allowed in CONFIG\_UNLOCK or ERROR state, although it is not explicit stated in TDISP specification, because there is no security property required.

#### 3.2.3. ATS Rule

The presence of DMA address translation in the host system has certain performance implications for DMA accesses. To mitigate these impacts, a device may include an address translation cache (ATC), which is also known as device translation look-aside buffer (Device

TLB). Address translation service (ATS) uses a request-completion protocol between a Device and a Root Complex (RC) to provide translation services. In addition, a new Address Type (AT) field is defined within the Memory Read and Memory Write TLP. The new AT field could be Untranslated, Translation Request, Translated.

ATS improves the behavior of DMA based data movement. An associated Page Request service (PRS) provides additional advantages by allowing DMA operations to be initiated without requiring that all the data to be moved into or out of system memory be pinned. Allowing a device to operate more independently (to page fault when it requires memory resources that are not present) provides a superior level of coupling between device and host.

ATS TLP Type definition:

- **Translated read/write:** Follow the same rule as Memory read/write for DMA/MMIO.
- **Translation Request (TRANS):** The translation request from the device Address Translation Cache (ATC) to the host Translation Agent (TA).
- **Translation Completion (TRANS-CPL):** The translation completion from the host TA to the device ATC.
- **Invalidate Request (INVAL):** The invalidate request from the host TA to the device ATC.
- **Invalidate Completion (INVAL-CPL):** The invalidate completion from the device ATC to the host TA.
- **Page Request (PAGE):** The page request from the device ATC to the host TA.
- **PRG Response (PGR-RSP):** The PRG response from the host TA to the device ATC.

Table 3-10, Table 3-11, and Table 3-12 are rule summary of ATS TLP, according to [PCIe TDISP 1.0] 11.4.10, page 53.

**Table 3-10: ATS Invalidate Request Rule Summary for TDI as Completer**

Access Control	CONFIG_UNLOCK	CONFIG_LOCK	RUN	ERROR
TEE-TLP	N/A	INVAL (V)	INVAL (V)	INVAL (V)
Non-TEE-TLP	INVAL (V)	INVAL (V)	INVAL (V)	INVAL (V)

1. **INVAL** rule: Invalidation Request is allowed in TEE-TLP or Non-TEE-TLP.

2. **INVAL-CPL** rule: Invalidation Completion must use the same IDE Stream as the Invalidation Request, and **must match** the T bit value from the Invalidation Request.

**Table 3-11: ATS Translation Request Rule Summary for TDI as Requester**

Operation	CONFIG_UNLOCK	CONFIG_LOCK	RUN	ERROR
TEE-TLP	N/A	TRANS (X)	TRANS (V)	TRANS (X)
Non-TEE-TLP	TRANS (V) **	TRANS (X)	TRANS (X)	TRANS (X)

\*\* TRANS is allowed in CONFIG\_UNLOCK, if the TDI is designed to a Legacy VM.

Table 3-12: ATS Page Request Rule Summary for TDI as Requester

Operation	CONFIG_UNLOCK	CONFIG_LOCK	RUN	ERROR
TEE-TLP	N/A	PAGE (X)	PAGE (V)	PAGE (X)
Non-TEE-TLP	PAGE (V) **	PAGE (X)	PAGE (X)	PAGE (X)

\*\* PAGE is allowed in CONFIG\_UNLOCK, if the TDI is designed to a Legacy VM.

1. **TRANS** rule: Translation Request is only allowed with T-bit set and in RUN state. Although it is not explicitly stated in TDISP specification, Translation Request is not allowed in other states (CONFIG\_UNLOCK or ERROR) or without T-bit set.
2. **TRANS-CPL** rule: Translation Completion(s) received with the T bit Clear must transition the TDI to **ERROR**.
3. **PAGE** rule: Page Request is only allowed with T-bit set and in RUN state.
4. **PGR-RSP** rule: A PRG Response must use the same IDE Stream as the corresponding Page Request and must have the T bit Set. A violation of this rule must result in the TDI transitioning to **ERROR**.

### 3.2.4. Peer to Peer (P2P)

PCIe peer to peer (P2P) enables two PCIe device endpoints (EPs) to transfer data between each other without using host memory as temporary storage. There are two types of P2P. See figure 3-3.

- **Direct P2P:** Two endpoints set up a dedicated P2P IDE stream with TDISP BIND\_P2P\_STREAM message. This feature requires the ATS is support and enabled for the device.
- **P2P via Root Complex:** Two endpoints set up two different IDE streams with Root Complex. If EP1 needs to send TLP to EP2, Root Complex will decrypt TLP in IDE stream 1 and encrypt it in IDE stream 2.

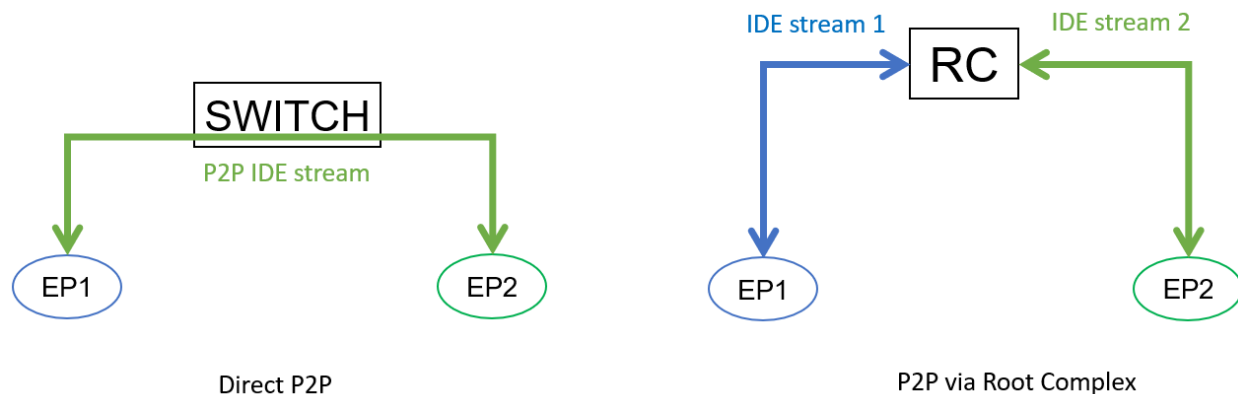


Figure 3-3: P2P types

The TLP rule for “P2P via Root Complex” is same as the TLP rule for TEE-MMIO (Table 3-4), Non-TEE-MMIO (Table 3-5), DMA (Table 3-7).

For TDX Connect compatibility, please refer to Appendix A “Access Control in TDX Connect” section.

### 3.3. TDISP Interoperability with PCIe Capabilities

#### 3.3.1. MSI-X

TDISP provides an optional support for TDIs to lock the MSI-X table the Pending Bit Array (PBA) in order enable trusted interrupts to TVM sending MSI-X requests using IDE-TLPs.

##### **Requirement if implemented:**

3.3.1. The device shall expose the MSI-X Capability register to the host.

3.3.2. The DSM shall lock the MSI-X table and PBA if indicated by the LOCK\_INTERFACE\_REQUEST\_FLAGS (Bit 2 – LOCK\_MSIX).

3.3.3. The DSM shall report the MSI-X capability message control register state in TDI Report MSI\_X\_MESSAGE\_CONTROL, according to [PCIe TDISP 1.0] Table 15 TDI Report Structure, page 37.

#### 3.3.2. ATS

ATS is an optional feature, this document does not cover TDISP with ATS enabled devices. For more details, please refer to [PCIe TDISP 1.0].

#### 3.3.3. Direct P2P

TDISP provides an optional mechanism to configure direct TDI P2P instead of using “P2P via Root Complex”. This document does not cover TDISP with direct P2P enabled devices. For more details, please refer to [PCIe TDISP 1.0].

#### 3.3.4. PASID

PASID is an optional feature that enables sharing of a single Endpoint device across multiple processes while providing each process a complete virtual address space. This document does not cover TDISP with ATS enabled devices. For more details, please refer to [PCIe TDISP 1.0].

#### 3.3.5. LNR

LNR is deprecated in PCRE 6.0, this document does not cover TDISP with LNR enabled devices. For more details, please refer to [PCIe TDISP 1.0].

#### 3.3.6. TPH

TLP Processing Hints (TPH) is an optional feature that provides hints in Request TLP headers to facilitate optimized processing of Requests that target Memory Space. This document does not cover TDISP with TPH enabled devices. For more details, please refer to [PCIe TDISP 1.0].

For TDX Connect compatibility, please refer to Appendix A “TDISP” section.

## 4 Device Security Architecture

This chapter describes the device security architecture requirement.

### 4.1. Resource Isolation and Protection

DSM shall implement access control and isolation mechanisms:

**Requirement:**

4.1.1. The device shall isolate the data for TDI from non-TDI.

4.1.2. The device shall isolate the data for one TDI from other TDIs.

4.1.3. The device shall scrub the confidential information in case of TDISP STOP\_INTERFACE, data integrity error, conventional reset, function level reset, according to [PCIe TDISP 1.0] 11.3.16. STOP\_INTERFACE\_REQUEST, 11.5.6. Data Integrity Errors, 11.4.8. Conventional Reset, 11.4.9. Function Level Reset.

4.1.4. The device should follow good practice including: securing secrets through the use of local encryption, access control, and/or other mechanisms; ensuring that secure data cannot “leak” due to errors, power management, or other operations; ensuring that secret keys are never exposed or stored in non-secure buffers; ensuring that the establishment & management of TEEs is itself secure, according to [PCIe IDE 1.0] 6.99 Implementation Note.

### 4.2. Address Translation

Device may implement ATS.

**Requirement:**

4.2.1. TEE-I/O capable devices must enforce integrity of the Address Translation Cache (ATC) such that the translations provided by the Root Complex cannot be modified through untrusted access. [PCIe TDISP 1.0] 11.4.10, page 54.

### 4.3. Device Resource

**Requirement:**

4.3.1. The DSM shall not support I/O resource for TVM, according to [PCIe TDISP 1.0] 11.2 TDISP Rules, page 20.

### 4.4. Device Identity and Measurement Reporting

DSM shall implement root of trust (ROT) for device attestation.

**Requirement:**

- 4.4.1. The device shall support device identity and authentication, according to [PCIe TDISP 1.0] 11.4.1 Device Identity and Authentication, page 49.
- 4.4.2. The device shall support firmware and configuration measurements, according to [PCIe TDISP 1.0] 11.4.2 Firmware and Configuration Measurement, page 49.
- 4.4.3. The DSM shall provide ROT for storage (RTS) to provide confidentiality for the device private key.
- 4.4.4. The DSM shall provide ROT for measurement (RTM) to record the measurement data at runtime.
- 4.4.5. The DSM shall provide ROT for reporting (RTR) to report the measurement data.
- 4.4.6. The device should provision the device certificate at manufacture time. A DICE device shall generate alias certificates at boot time.
- 4.4.7. The Device vendor should publish the reference integrity manifest (RIM) for attestation. The RIM may follow [IETF-CORIM], [DICE Endorsement] or [TCG Comp RIM IM], etc.

#### 4.4.1. Device Firmware Resilience

Usually, the device supports firmware update. The DSM shall implement ROT for resilience.

**Requirement:**

- 4.4.8. The DSM shall provide ROT for update (RTU) for secure firmware update, including update image integrity protection and rollback protection, according to [NIST SP 800-193].
- 4.4.9. The DSM shall provide ROT for detection (RTD) for secure boot, including boot image integrity verification and secure version number (SVN) verification, according to [NIST SP 800-193].
- 4.4.10. The DSM should provide ROT for recovery (RTRec) to recover the device firmware in case of verification failure, according to [NIST SP 800-193].

#### 4.4.2. Runtime Firmware Update

Optionally, the device may support runtime update without reset. The capability is controlled by following fields:

- **SPDM 1.2 SessionPolicy.TerminationPolicy:** Determine if runtime update will keep session alive.
- **TDISP LOCK\_INTERFACE\_REQUEST.NO\_FW\_UPDATE:** Determine if update is allowed in TDISP CONFIG\_LOCKED state.

See table 4-1.

Table 4-1: Runtime Firmware Update Summary

SPDM: TerminationPolicy	TDISP: NO_FW_UPDATE	Result
0 (No Runtime Update)	0 (Allow firmware update if TDISP is in CONFIG_LOCKED state)	No Runtime Update.  The update will cause SPDM session termination. After update, the device will expect next SPDM message to be GET_VERSION, and will return ERROR(RequestResynch) for all other.
0 (No Runtime Update)	1 (Not allow firmware update if TDISP is in CONFIG_LOCKED state)	No Runtime Update.  Any update is not allowed if TDISP is in CONFIG_LOCKED state. The update is allowed if TDISP is not in CONFIG_LOCKED state and the update will cause SPDM session termination.
1 (May support Runtime Update)	0 (Allow firmware update if TDISP is in CONFIG_LOCKED state)	The device may choose to allow Runtime Update regardless of the TDISP state.  The update may keep session alive or terminate. It is the device's choice based on the impact of the update.
1 (May support Runtime Update)	1 (No allow firmware update if TDISP is in CONFIG_LOCKED state)	The device may choose to allow Runtime Update if TDISP is not in CONFIG_LOCKED state.  Any update is not allowed if TDISP is in CONFIG_LOCKED state.

**Requirement:**

- 4.4.11. The device shall support SPDM 1.2 SessionPolicy.TerminationPolicy = 1, to keep the SPDM session alive during runtime update.
- 4.4.12. The device shall keep IDE stream alive during the runtime update.
- 4.4.13. The device shall support TDISP LOCK\_INTERFACE\_REQUEST.NO\_FW\_UPDATE = 0, to keep TDISP alive during the runtime update.
- 4.4.14. The device shall support SPDM\_DIGESTS and SPDM\_CERTIFICATE command in Session.
- 4.4.15. The device should support SPDM 1.2 MEAS\_FRESH\_CAP to report the fresh measurement after the runtime update.

## 4 Device Security Architecture

4.4.16. The device should support SPDM 1.2, MEASUREMENTS.Param2.content\_change detection to report the atomicity of the measurement reporting.

4.4.17. The device should support SPDM 1.2, DMTFSpecMeasurementValueType Mutable Firmware Security Version Number (SVN).

4.4.18. The DICE device shall report the certificate including the new firmware information in DiceTcbInfo, such as firmware digest and/or SVN.

4.4.19. The DICE device should report SPDM defined Hardware Identity OID in the SPDM certificate chain to identify the hardware identity, if the DICE certificate chain is changed during runtime firmware update.

## 4.5. Secure Interconnects

The device shall support IDE based secure communication with the TSM.

### Requirement:

4.5.1. The devices must support selective IDE. [PCIe TDISP 1.0] 11.4.3 Secure Interconnects, page 50.

4.5.2. The device must implement adequate security measures to prevent leakage of the encryption key at rest and in use, according to [PCIe TDISP 1.0] 11.4.3 Secure Interconnects, page 50.

## 4.6. Device Attached Memory

A device may implement device attached memory, which is used to host the TVM data.

### Requirement:

4.6.1. If device attached memory is supported, the device shall ensure the confidentiality and optional integrity of the TVM data stored in the device attached memory, according to [PCIe TDISP 1.0] 11.4.4 Device Attached Memory, page 50.

## 4.7. TDI Security

The device shall support TDISP protocol to manage TDI state.

### Requirement:

4.7.1. The device shall follow TDI Security requirement, according to [PCIe TDISP 1.0] 11.4.5 TDI Security, page 51. The device shall support TDI state (CONFIG\_UNLOCK, CONFIG\_LOCK, RUN, ERROR) and IDE Stream state (Insecure, Secure) transition. Any configuration change that

impacts the TDI security properties shall result in the TDI ERROR state and IDE Stream Insecure state.

## 4.8. Data Integrity Errors

A device may receive a poisoned TLP on an interface in RUN.

### Requirement:

4.8.1. The device shall handle data integrity error according to [PCIe TDISP 1.0] 11.4.6 Data Integrity Error, page 52. The device shall change the interface from RUN to ERROR if a poisoned TLP is received, to prevent bad data consumption and propagation. The device shall scrub and clear information in such logs and reporting registers (e.g., syndrome) that may reveal confidential data.

## 4.9. Debug Modes

A device may support multiple debug modes or debug capabilities.

### Requirement:

4.9.1. The device shall handle the debug mode, according to [PCIe TDISP 1.0] 11.4.7 Debug Modes, page 52. Debug capabilities must not affect the security of the device and must not lead to a compromise of the confidentiality or integrity of the TVM data provided to the device.

### 4.9.1. Device Debug Interface

A device may provide debug interface to access low level data.

### Requirement:

4.9.2. The device shall implement the debug interface without impacting on the security properties.

4.9.3. The device should report the debug state to the host if the debug mode is enabled or a debugger is attached. The mechanism may be in SPDM measurement DeviceModeCapabilities or DiceTcbInfo flags.

## 4.10. Device Reset

### 4.10.1. Conventional Reset

A conventional reset (cold, warm, or hot) leads to the device changing all its port registers and state machines to their initialization values, and the TDISP state of all TDIs transitions to CONFIG\_UNLOCKED.

**Requirement:**

4.10.1. The device shall handle the conventional reset, according to [PCIe TDISP 1.0] 11.4.8 Conventional Reset, page 53. The device shall ensure that all TVM data, IDE keys, other encryption keys (e.g., P2P links, intra-device interconnects, etc.) and SPDM session keys are cleared and not exposed in conventional reset. The device shall reset the device measurement registers to their default values in conventional reset.

4.10.2. The device shall handle the conventional reset, according to [PCIe IDE 1.0] 6.99.8 Other IDE Rules. Any Conventional Reset to an Upstream Port or to the Bridge Function of a Downstream Port must result in all IDE Streams associated with that Function transitioning to the Insecure state, and all keys must be invalidated and rendered unreadable.

### 4.10.2. Function Level Reset (FLR)

A device may support function level reset.

**Requirement:**

4.10.3. The device shall handle the Function Level Reset (FLR), according to [PCIe TDISP 1.0] 11.4.9 Function Level Reset, page 53. The device shall ensure that all affected TDI from CONFIG\_LOCKED, RUN state to ERROR state in function level reset. As such, the host needs to issue STOP\_INTERFACE\_REQUEST to clean up the TDI state and scrub TVM data/secrets.

4.10.4. The device shall handle the Function Level Reset (FLR), according to [PCIe IDE 1.0] 6.99.8 Other IDE Rules. Any FLR to a Function containing an IDE Extended Capability must result in all IDE Streams associated with that Function transitioning to the Insecure state, and all keys must be invalidated and rendered unreadable. An FLR to a Function that does not contain an IDE Extended Capability must not affect IDE operation.

Table 4-2 shows examples of the impact of different reset, assuming that Physical Function FLR (PF-FLR) happens on a function that contains an IDE Extended Capability and Virtual Function FLR (VF-FLR) happens on a function that does not contain an IDE Extended Capability.

**Table 4-2: Examples of the Impact of Reset**

Reset Type	VF specific TDI	Other subordinate VF TDIs	IDE Stream	SPDM Session
------------	-----------------	---------------------------	------------	--------------

Conventional Reset	ERROR	ERROR	Insecure	Termination
PF-FLR (with IDE ECAP)	ERROR	ERROR	Insecure	No impact
VF-FLR (no IDE ECAP)	ERROR	No impact	No impact	No impact

## 4.11. Timing

A device shall follow the timing requirement defined in the standards. See table 4-3.

Table 4-3: Timing requirement summary

Protocol	Description
PCI DOE	The device shall return DOE response within <b>1 second</b> , according to [PCIe DOE 1.0] 6.xx.1 Operation.
SPDM	The device shall follow “ <b>Timing specification table</b> ” and the CTE exponent shall be returned via SPDM CAPABILITIES.
IDE	<ol style="list-style-type: none"> <li>The device port shall be able to process IDE TLP within <b>10ms</b> after receiving the IDE_KM K_SET_GO (enable or refresh), according to [PCIe IDE 1.0].</li> <li>The device port shall invalidate and render unreadable the key set within <b>10ms</b> after receiving the IDE_KM K_SET_STOP, according to [PCIe IDE 1.0].</li> <li>When aggregating TLPs, the Transmitter must treat a TLP as the last TLP of an aggregated unit unless the Transmitter can guarantee that it will transmit another TLP within the aggregated unit within <b>1µs</b>.</li> </ol>
TDISP	The device shall inherit timing requirements from the SPDM.
TLP	The device shall follow [PCIe 6.0] 2.8 Completion Timeout Mechanism

## 4.12. Error Handling

A device shall follow the error handling requirement defined in the standards.

### 4.12.1. Error Trigger

Table 4-4 shows the possible source that triggers the error.

Table 4-4: Error Trigger

Source	IDE Insecure	TDISP ERROR
DOE Mailbox	If <b>DOE mailbox error causes unrecoverable error and SPDM session termination</b> , then IDE state shall be changed to insecure.	If <b>DOE mailbox error causes unrecoverable error and SPDM session termination</b> , then TDI state shall be changed to ERROR.
SPDM Session	<b>SPDM session termination</b> shall cause the IDE state to insecure.	<b>SPDM session termination</b> shall cause the TDI state to ERROR. [PCIe TDISP 1.0] 11.4.5. TDI Security.

	<p>[PCIe TDISP 1.0] 11.4.5. TDI Security.</p> <p>If the secure SPDM session that was used for <b>initial key programming is closed, any subsequent QUERY and/or KEY_PROG requests received through a different secure SPDM session</b> must first cause the responder to invalidate and render unreadable all keys must for the IDE Stream, then transition that IDE Stream to the Insecure state. [PCIe IDE 1.0] 6.99.3. IDE KM. Page 24.</p>	
TLP / Configuration	<ol style="list-style-type: none"> <li>Any <b>conventional reset</b> or any <b>FLR to a function containing an IDE extended capability</b> must result in IDE stream to insecure state. [PCIe IDE 1.0] 6.99.8 Other IDE Rules. <i>NOTE: An FLR to a Function that <b>does not contain an IDE Extended Capability</b> must <b>not</b> affect IDE operation.</i></li> <li>Use of Selective IDE under any use case where ACS services (or any other mechanism) <b>blocks</b> or otherwise <b>terminates IDE TLPs</b> will result in the associated Selective IDE Stream going to Insecure. [PCIe IDE 1.0] 6.99.8 Other IDE Rules.</li> <li>Using the following practices. [PCIe IDE 1.0] 6.99 Integrity &amp; Data Encryption (IDE) implementation note. <ol style="list-style-type: none"> <li><b>Detecting inappropriate attempts to reconfigure IDE,</b> and/or other internal conditions that could compromise secure data forcing the Port into Insecure.</li> </ol> </li> </ol>	<ol style="list-style-type: none"> <li>Following conditions must be treated as errors, according to [PCIe TDISP 1.0] 11.2 TDISP Rules page 19. <ol style="list-style-type: none"> <li>Changes to TDI configuration that <b>affect the configuration or the security</b> of the TDI. [PCIe TDISP 1.0] 11.2.6. <b>DSM Tracking and Handling of Locked TDI</b>, 11.4.5. TDI Security. Refer to Table 2: Example DSM Tracking and Handling for Architected Registers.</li> <li><b>Changes to the Requester ID</b></li> <li>Resetting the TDI using a <b>Function Level Reset</b>.</li> <li>Any IDE stream bound to the TDI transitions to the <b>Insecure state</b>. [PCIe TDISP 1.0] 11.4.5. TDI Security.</li> <li>Receipt of a <b>poisoned TLP</b> ([PCIE Base] 6.2.3.3 Error Forwarding) or detecting data integrity errors in the device for data associated with that TDI, where the error is not recoverable.</li> <li>Other device specific conditions or <b>changes in configuration that affect trust</b> properties.</li> </ol> </li> </ol>

	<p>3.2. Any <b>change in debug configuration</b> that could expose data intended to be secured result in a transition to Insecure.</p>	<p>2. Receipt of a <b>Completion with UR/CA or Completion timeout</b> (following recovery retries) for request initiated by a TDI in <b>CONFIG_LOCK, RUN</b> (with T=1) indicates occurrence of an uncorrectable error. [PCIe TDISP 1.0] 11.4.3. Securing Interconnects.</p> <p>3. ATS Error [PCIe TDISP 1.0] 11.4.10</p> <p>3.1. <b>Translation Completion(s)</b> received with the <b>T bit Clear</b>, if the request with T bit set.</p> <p>3.2. <b>PRG Response</b> received with the <b>T bit clear</b>, if the request with T bit set.</p>
--	--	--

Table 4-5 shows the IDE TLP Error. “IDE Check Failed” is a fatal error and will cause the IDE state to be Insecure. “Misrouted IDE TLP” and “PCRC Check Failed” are non-fatal error and will not cause an IDE state change.

**Table 4-5: IDE TLP Error**

IDE Error	Error Condition
IDE Check Failed	<p>1. Selective IDE Stream rules which cause an IDE Check Failed error. [PCIe IDE 1.0] 6.99.4 IDE TLPs</p> <p>1.1. Receipt of an IDE TLP associated with a Selective IDE Stream that is <b>not a permitted TLP Type</b>.</p> <p>2. Aggregation TLP rules that cause an IDE Check Failed error. [PCIe IDE 1.0] 6.99.6 IDE TLP Aggregation</p> <p>2.1. If the K bit is to be toggled, it must only be toggled for the <b>first TLP of an aggregated unit</b>.</p> <p>2.2. If an IDE TLP with the <b>M bit Clear</b> is received at a Receiver where <b>Aggregation is not supported</b>, or if nine or more successive TLPs are received in the Sub-Stream with the M bit Clear.</p> <p>3. Other rules that cause an IDE Check Failed. [PCIe IDE 1.0] 6.99.8 Other IDE Rules.</p> <p>3.1. Received Completion shall use <b>same Stream ID and Same T bit</b> with NPR.</p> <p>3.2. Use of mechanisms that result in the blocking or termination of TLPs must be carefully coordinated with the use of Selective IDE Streams.</p> <p><b>Dropping of Selective IDE TLPs.</b></p>

	<p>3.3. Detection following condition:</p> <p>3.3.1 <b>MAC check failure</b></p> <p>3.3.2 <b>underflow</b> of PR-received-counter-NPR/CPL</p> <p>3.3.3 <b>overflow</b> of PR-received-counter-NPR/CPL</p> <p>3.3.4. <b>unsupported field</b> in sub-stream identifier. [PCIe Base] 6.33.8 Other IDE Rules.</p>
Misrouted IDE TLP	<p>1. Flow-Through selective ID stream rules which cause Misrouted IDE TLP. [PCIe TDISP 1.0] 6.99.7 Flow-Through Selective IDE Streams</p> <p>1.1 If an IDE TLP is routed to an Egress Port with the Flow-Through IDE Stream Enabled bit Clear.</p> <p>2. Other IDE rules that cause Misrouted IDE TLP. [PCIe TDISP 1.0] 6.99.8 Other IDE Rules</p> <p>2.1. Receipt of a Link IDE TLP or Selective IDE TLP for which there is not an associated IDE Stream</p> <p>2.2. Receipt of a Link IDE TLP by a Switch that targets an Egress Port for which there is not a Link IDE Stream associated with the same TC and in the Secure state</p>
PCRC Check Failed	<p>1. IDE TLP rules that cause PCRC Check Failed. [PCIe TDISP 1.0] 6.99.4 IDE TLPs</p> <p>1.1. When PCRC is enabled for an IDE Stream, the PCRC is not present.</p> <p>1.2. The PCRC must only be checked by the ultimate Receiver of the IDE TLP including PCRC. A failure of the PCRC check indicates that one or more bits of the data payload have been corrupted.</p>

#### 4.12.2. Error Notification

Table 4-5 shows the error notification via software protocol.

**Table 4-6: Error Notification via Protocol**

Source	Protocol Error
DOE Message	N/A
SPDM	SPDM response may return error via <b>SPDM_ERROR</b> .
IDE_KM	IDE_KM response may return <b>error via Status Field</b> in IDE_KM KP_ACK, according to [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE_KM). Attempting to configure IDE keys into a sub-stream using different SPDM sessions is an error and must be rejected. [PCIe TDISP 1.0] 11.4.5. TDI Security.
TDISP	TDISP response may return error via <b>TDISP_ERROR</b> , according to [PCIe TDISP 1.0] 11.3.8. LOCK_INTERFACE_REQUEST, 11.3.10. GET_DEVICE_INTERFACE_REPORT, 11.3.12. GET_DEVICE_INTERFACE_STATE, 11.3.14. START_INTERFACE_REQUEST, 11.3.16. STOP_INTERFACE_REQUEST, 11.3.18. BIND_P2P_STREAM_REQUEST, 11.3.20. UNBIND_P2P_STREAM_REQUEST, 11.3.22. SET_MMIO_ATTRIBUTE_REQUEST.

TLP	<p>TLP may return Unsupported Request (UR) or Unexpected Completion (UC), according to [PCIE Base] 6.33.8 Other IDE Rules.</p> <p>The Translation <b>Complete Status field</b> is defined in [PCIE Base] Table 10-2 Translation Completion with No Data Status Codes.</p> <p>The transaction layer Error is listed at [PCIE Base] Table 6-5 Transaction Layer Error List.</p>
-----	---

Table 4-7 shows the error notification via error register.

**Table 4-7: Error Notification via Register**

Source	Register
DOE Error	<b>DOE Error</b> (bit 2) is in [PCIe DOE 1.0] 7.9.24.4 DOE Status Register.
IDE Error	<p>Link IDE <b>Integrity Check Fail</b> (bit 31) is in [PCIe IDE 1.0] 7.9.99.4.2 Link IDE Stream Status Register.</p> <p>Selective IDE <b>Integrity Check Fail</b> (bit 31) is in [PCIe IDE 1.0] 7.9.99.5.3 Selective IDE Stream Status Register.</p> <p>The <b>Integrity Check Fail</b> is equal to <b>IDE Check Failed. Misrouted IDE TLP or PCRC Check Failed</b> error shall not cause <b>Integrity Check Fail</b> error bit set.</p>
Advanced Error (optional)	<p>Unrecoverable Error Status Register is defined in [PCIe IDE 1.0] 7.8.4 Advanced Error Reporting Extended Capability, including <b>IDE Check Failed</b> (bit 28), <b>Misrouted IDE TLP</b> (bit 29), <b>PCRC Check Failed</b> (bit 30).</p>

NOTE: There is no proactive TDISP error notification via either a hardware or software mechanism. The reason is that there is no way to guarantee that such proactive notification will be delivered to a TVM when the error happens, such as an MSI-X interrupt or TDISP error event message. Even if a device chooses to implement such a notification mechanism, it is not secured because the man-in-the-middle adversary may block or delay the error notification. Before the TVM receives such notification, it may already consume the invalid data. As such, when error happens, the only requirement from the device side is to stop working immediately.

When TVM performs MMIO\_READ action, it will usually get All-1 (such as 0xFF for 8bit reg, 0xFFFF for 16bit reg, or 0xFFFFFFFF for 32-bit reg) if the device is in ERROR state. The best practice is for the device to design the MMIO register in a way that All-1 can be considered an error. As such, when TVM gets All-1, it knows that device is in the ERROR state. Also, the TVM may use GET\_DEVICE\_INTERFACE\_STATE to poll the TDI state to confirm whenever it suspects TDI being in ERROR state.

## 4.12.3. Error Recovery

Table 4-8 shows the possible error recovery mechanism.

Table 4-8: Error Recovery

Source	Recovery
DOE	If DOE Error (bit 2) in DOE Status Register is set, the device shall wait for host software to set DOE Abort (bit 0) in DOE Control Register to clear the Error.
SPDM	If there is SPDM session error or heartbeat timeout, the device shall terminate the session and wait for new <b>KEY_EXCHANGE</b> message to set up a new SPDM session, or terminate the SPDM connection and wait for <b>GET_VERSION</b> message to set up a new SPDM connection.
IDE	<ol style="list-style-type: none"> <li>Once <b>IDE Check Failed</b> error is detected, the TLP that triggered the error and all subsequent IDE TLPs received associated with the same IDE Stream must be discarded, according to [PCIe IDE 1.0] 6.99.8 Other IDE Rules, page 45. <ol style="list-style-type: none"> <li>For <b>Link IDE (TC0/VC0)</b>, it will be impossible to communicate with the device until <b>the device is reset</b>, e.g. using a Secondary Bus Reset (a PCIe Hot Reset), because the device must reject all subsequent coming TLPs.</li> <li>For <b>Selective IDE</b>, if Configuration Requests are not being associated with the Stream, then it is possible to <b>recover the device by tearing down the affected TDIs, disabling and then re-enabling the Selective IDE Stream, and then restarting the TDIs</b>.</li> </ol> </li> <li><b>Misrouted IDE TLP</b> is not a fatal error. The device shall permit continued operation. [PCIe IDE 1.0] Table 605.</li> <li><b>PCRC Check Failed</b> is not a fatal error. The device shall permit continued operation. [PCIe IDE 1.0] Table 605.</li> </ol>
TDISP	<ol style="list-style-type: none"> <li>For TDI ERROR, the host can <b>send STOP_INTERFACE_REQUEST</b> to change the TDI to CONFIG_UNLOCKED.</li> <li>A TDI is permitted to <b>transport TLP messages in ERROR state if and only if T bit is set</b>, e.g. ATS invalidate. [PCIe TDISP 1.0] 11.2.1 TLP Rules, page 21.</li> <li><b>A TDI is permitted so that clearing this data be deferred</b> until the receipt of a STOP_INTERFACE_REQUEST to transition the TDI to CONFIG_UNLOCKED. [PCIe TDISP 1.0] 11.2.1 TLP Rules, page 19.</li> <li>It is permitted that the TDI transition <b>automatically from ERROR to CONFIG_UNLOCKED</b>, if and only if the TDI first clears all TVM confidential data. [PCIe TDISP 1.0] 11.2.1 TLP Rules, page 19.</li> </ol>

NOTE: Once TDI is in ERROR state, the TDI shall clear all security related context. Care must be taken that the Tag in NPR without CPL shall not in the context. The TDI shall keep tracking the CPL Tag and ensure the same Tag is never reused when the TDI is out of ERROR state to avoid Tag reuse attack. Alternatively, the TDI may disallow change from ERROR to other state before

#### 4 *Device Security Architecture*

the TDI receives a corresponding CPL TLP or CPL timeout. Or the TDI can do reset such as Function Level Reset (FLR) or convention reset.

## 5 Summary

This white paper describes how to build a device to support confidential computing. First, we provide a summary of the secure device interface lifecycle. Then we provide detailed information on software communication (DOE, SPDM, IDE\_KM, TDISP) and the hardware communication (link encryption), as well as the device security implementation.

## Appendix A: Intel® TDX Connect Interoperability

This section describes the features supported in Intel® TDX Connect host architecture and restrictions that are introduced to the generic device architecture described in the main portion of this document.

Table A-1 shows the generic terminology mapping for Intel® TDX Connect.

**Table A-1: Terminology Mapping for Intel TDX**

Term	Intel® TDX Connect
TEE Security Manager (TSM)	Intel® TDX Module
TEE Virtual Machine (TVM)	Tenant Trust Domain (TD)

In following sections, we will use term “Intel TSM” to indicate the TSM in Intel TDX Connect, including Intel® TDX Module.

### A.1. TDX Connect Software Interoperability

This section describes the software stack requirements for device interoperability with the TDX Connect host.

#### A.1.1. DOE

The DOE functionality required to be supported by the TDX Connect device is:

- DOE Discovery
- CMA/SPDM
- Secure CMA/SPDM

The following DOE functionality is not supported by the TDX host, and the device shall not use them for any TDX Connect usages.

- CMA/SPDM with Connection ID
- Secure CMA/SPDM with Connection ID
- Async Message

#### Configurations

Depending on the device’s architecture and functionality, a device shall implement one or more DOE mailboxes. The mailboxes may be in the following configurations:

- 1 device -> N functions -> 1 DOE (in function 0 only)
- 1 device -> N functions -> N DOEs
- 1 device -> N functions -> N\*M DOEs (each function has M DOEs)

The TDX host does not place any restrictions on the number of DOE mailboxes and how they are associated with the functions other than overall platform capacity.

### A.1.2. SPDM

TDX Connect host supports SPDM version 1.2.

The device shall support SPDM version 1.2 or higher. If the device supports a higher version than 1.2, it shall be able to fall back to using version 1.2 for TDX Connect usages through the SPDM Get Capabilities mechanics.

The device shall support the SPDM request and response messages in table A-2. “Optional” means the TDX Connect host may send the message if the device supports the capability. “In session” means the TDX Connect host needs to send the message in SPDM session. “In/out of session” means the TDX Connect host needs to send the message in SPDM session and out of SPDM session.

**Table A-2: Supported SPDM Messages**

<b><i>Request Messages</i></b>	<b><i>Response Messages</i></b>
GET_VERSION	VERSION
GET_CAPABILITIES	CAPABILITIES
NEGOTIATE_ALGORITHMS	ALGORITHMS
GET_DIGESTS (in/out of session)	DIGESTS (in/out of session)
GET_CERTIFICATE (in/out of session)	CERTIFICATE (in/out of session)
GET_MEASUREMENTS (in session)	MEASUREMENTS (in session)
KEY_EXCHANGE	KEY_EXCHANGE_RSP
FINISH	FINISH_RSP
<i>HEARTBEAT (optional)</i>	<i>HEARTBEAT_ACK (optional)</i>
<i>KEY_UPDATE (optional)</i>	<i>KEY_UPDATE_ACK (optional)</i>
END_SESSION	END_SESSION_ACK
-	ERROR

### ***DOE mappings***

SPDM may be mapped to the device/function and DOEs in the following ways.

- 1 device -> N functions -> 1 DOE (in function 0 only) -> 1 SPDM.
- 1 device -> N functions -> N DOEs -> N SPDMs
- 1 device -> N functions -> N DOEs -> 1 SPDM (other N-1 DOE is used for other purposes)
- 1 device -> N functions -> N\*M DOEs (each function has M DOE) -> N SPDMs

VMM will discover and select the proper DOE mailbox one that supports IDE\_KM and TDISP. Then the VMM will ask TDX-Module to set up the SPDM session.

### ***Cryptographic algorithms***

The TDX Connect host supports the SPDM cryptographic algorithms defined in CMA ECN.

The device may implement any subset of the algorithms governed by those specifications.

### Mutual Authentication

The TDX Connect host does not support mutual authentication. The host will not set MUT\_AUTH\_CAP. The DSM shall not request mutual authentication.

The device may optionally implement a non-standardized mutual authentication-attestation method as mentioned in chapter 2, such as RATLS.

### Certificate

Intel TSM is the host component that will receive the certificate chain from the device, verify the signature by using the public key associated with the leaf certificate of the Responder, and all intermediate public keys within the certificate chain using the root certificate as the trusted anchor. It will use the public key associated with the leaf certificate to set up the SPDM session.

Intel TSM provides a secure mechanism to pass the entire certificate chain in all slots to the TVM, so that the TVM can verify its contents and make decisions to accept the device using the TVM's own policy.

### Measurement

Intel TSM will receive the device measurements from the device in the SPDM session.

Intel TSM provides a secure mechanism to pass all measurements received from the device to the TVM, so that the TVM can verify the device measurements based TVM's policy.

### SPDM Connection and Session

The device may support multiple SPDM connections and multiple SPDM sessions in one SPDM connection.

- 1 DOE mailbox -> 1 Connection -> 1 Session
- 1 DOE mailbox -> X Connections -> X Sessions
- 1 DOE mailbox -> X Connections -> X\*Y Sessions

Intel TSM will only choose 1 SPDM Connection and setup 1 SPDM session.

#### A.1.3. IDE\_KM

The TDX Connect device shall support the IDE\_KM defined in [PCIe IDE 1.0].

The device shall support the IDE\_KM request and response messages in table A-3.

Table A-3: Supported IDE\_KM Messages

<b>Request Messages</b>	<b>Response Messages</b>
QUERY	QUERY_RESP
KEY_PROG	KP_ACK
K_SET_GO	K_GOSTOP_ACK
K_SET_STOP	K_GOSTOP_ACK

### Selective IDE Stream Sequence

The TDX Connect host supports the following IDE key management sequence.

- Initial IDE Stream setup. The TDX Connect host SOC only supports the sequence to set IDE Stream Enable bit for SOC **after** the IDE key and the IDE key set are programmed into SOC Root Port and the device.
- IDE Stream Stop. The TDX Connect host SOC only supports the sequence to disable IDE stream Enable bit for device and then for SOC, before sending IDE\_KM message to the device.
- IDE Stream Key refresh.

Please refer to Appendix B, IDE Stream section, for an example on how to program keys into device and start IDE stream.

#### A.1.4. TDISP

The TDX Connect device shall support [PCIe TDISP 1.0]. That is the only TDISP version supported by the TDX Connect host.

The device shall support the TDISP request and response messages in table A-4.

**Table A-4: Supported TDISP Messages**

<b>Request Messages</b>	<b>Response Messages</b>
GET_TDISP_VERSION	TDISP_VERSION
GET_TDISP_CAPABILITIES	TDISP_CAPABILITIES
LOCK_INTERFACE_REQUEST	LOCK_INTERFACE_RESPONSE
GET_DEVICE_INTERFACE_REPORT	INTERFACE_REPORT
GET_DEVICE_INTERFACE_STATE	GET_DEVICE_INTERFACE_STATE
START_INTERFACE_REQUEST	START_INTERFACE_RESPONSE
STOP_INTERFACE_REQUEST	STOP_INTERFACE_RESPONSE
-	TDISP_ERROR

Intel TSM only support above TDISP mandatory messages in Table A-4.

#### Device Address Width

The TDX Connect compliant device's address width must be at least 52 bits.

The device reports the address width through the TDISP GET\_TDISP\_CAPABILITIES exchange. Intel TSM will reject devices that do not support at least 52 bits.

#### TDI Report Structure

Once the device returns TDI Report structure to the host. The host software should check the TDI report to decide if the device configuration is acceptable.

Intel TSM will perform basic **TSM capability check** for the TD Report Structure. See Table A-5.

**Table A-5: Intel TSM Capability Check for TDI Report**

<b><i>TDI Report Structure Field</i></b>	<b><i>Intel TSM Capability</i></b>
INTERFACE_INFO:BIT0 (Firmware update not allowed in CONFIG_LOCKED or RUN)	Ignored
INTERFACE_INFO:BIT1 (TDI generates DMA requests without PASID)	Must be 1
INTERFACE_INFO:BIT2 (TDI generates DMA requests with PASID)	Must be 0
INTERFACE_INFO:BIT3 (ATS supported and enabled for the TDI)	Must be 0
INTERFACE_INFO:BIT4 (PRS supported and enabled for the TDI)	Must be 0
MSI_X_MESSAGE_CONTROL	Must be 0
LNR_CONTROL	Must be 0
TPH_CONTROL	Must be 0
MMIO_RANGE_COUNT	Ignored
MMIO_RANGE	Ignored
DEVICE_SPECIFIC_INFO_LEN	Ignored
DEVICE_SPECIFIC_INFO	Ignored

The TVM should perform **policy check**. For example, if firmware update is allowed in CONFIG\_LOCKED or RUN state.

The TDX Connect compliant device shall follow the TSM capability to return the TDI report structure.

## **A.2. TDX Connect Hardware Interoperability**

This section describes the hardware stack requirements for device interoperability with the TDX Connect host.

### **A.2.1. IDE Stream**

TDX Connect compliant host SOC only supports PCI Express IDE and not CXL IDE.

The TDX Connect compliant device shall follow the PCI Express device requirements for IDE.

#### ***Selective IDE Stream Support***

The TDX Connect host SOC only supports selective IDE streams. The SOC does not support Selective IDE for Configuration Requests Enable.

The TDX Connect device shall only use selective IDE streams to communicate with TDX Connect host without Configuration Requests Enable.

### Number of Selective IDE Streams Supported

The TDX Connect host supports a total of up to 4 IDE streams per Root-Complex. The actual number of IDE stream *register blocks per Root Port* (RP) depends on per RP bifurcation as defined in the following table.

The device can support 1~4 Selective IDE streams depending on the interface as defined in the table A-6.

**Table A-6: number of IDE stream register blocks per Root Port**

<b>Bifurcation</b>	<b>Selective IDE Register Blocks</b>	<b>Link IDE Register Blocks</b>
1x16	4	1
2x8	3	1
4x4	1	1
8x2	N/A	N/A
16x1	N/A	N/A

The TDX Connect host only supports 1 IDE Address Association Register block and 1 IDE RID Association Register block per each selective IDE stream.

### TLP MAC Aggregation

The TDX Connect host SOC does not support TLP MAC aggregation. Intel TSM will always select the No Aggregation mode in the Selective Stream Control Register.

### IDE Support for TEE-IO

The TDX Connect host SOC Root Port IDE is TEE-IO capable. However, it does not expose the TEE-IO Supported Bit in IDE Device Capability register in [PCIe TDISP 1.0].

Intel TSM requires the software to enumerate TEE-IO support and enable it using Intel TSM host VMM interface.

### TEE Limited Stream

A TDISP device may report TEE Limited Stream capability, however, Intel TDX Connect host platform may not support enabling it. Before platform software may enable TEE Limited Stream on the EP, it shall check for TDX module support.

### A.2.2. TDISP

The TDX Connect device shall support [PCIe TDISP 1.0]. That is the only TDISP version supported by the TDX Connect host.

### Access Control in TDX Connect

The Intel TSM does not have knowledge on the device's TDI state, because Intel TSM cannot guarantee how the DSM does the transition. A TDI may transition to the ERROR state directly if the DSM detects a security violation or function level reset (FLR). A TDI may transit to CONFIG\_UNLOCKED in case of conventional reset. A TDI may automatically transit from ERROR to CONFIG\_UNLOCKED if the TDI clears all TVM confidential data, according to [PCIe TDISP 1.0] page 19.

#### MMIO Access Control

Intel TSM does not have knowledge on which device MMIO region is TEE-MMIO or Non-TEE-MMIO. The device will return TEE-MMIO information via INTERFACE\_REPORT. Intel TSM does not parse the information but passes it to the TVM directly. As such, we need define the MMIO resource in a different way in TDX Connect host.

MMIO Resource definition:

- **Private MMIO:** The MMIO range whose GPA.S = 0 and HPA TDX HKID is the TD private HKID. *In Intel TDX Connect architecture, the private MMIO can only be high MMIO whose address is above 4GB (MMIO-H) and not PCI Express configuration space (CFG).* The low MMIO range whose address is below 4GB (MMIO-L) or CFG cannot be private MMIO. Private MMIO access is only allowed in TEE-TLP.
- **Shared MMIO:** All other MMIO which is not private MMIO, including PCI Express configuration space. Shared MMIO access is only allowed in Non-TEE-TLP.

Role and Responsibility:

- **VMM is the resource manager.** VMM should allocate MMIO region and assign them to the devices.
- **TVM is the policy maker.** TVM should parse the TEE-MMIO range from INTERFACE\_REPORT and fully understand which MMIO is TEE-MMIO and which is Non-TEE-MMIO. The TVM shall accept private GPA mapping from VMM according to the reported TEE-MMIO range.
- **Intel TSM is the policy enforcer.** Intel TSM provides API (TDG.MMIO.ACCEPT) for the TVM to let TVM accept the TEE-MMIO mapped by VMM as private GPA. Intel TSM will ensure that private GPA access uses TEE-TLP and shared GPA access uses Non-TEE-TLP. Intel TSM ensures the TEE-MMIO is mapped as private GPA before allowing the DSM to transit to RUN state. Since only MMIO-H can be private memory, Intel TSM will reject the private MMIO map request for MMIO-L or CFG.

The following table A-7 shows the MMIO Access Rules for TDX Connect host as Requester. For TEE-TLP, Intel TSM will ensure that only TVM (as TDI owner) can generate such access.

Table A-7: MMIO Rule for TDX Connect host as Requester

Operation	TEE-TLP (Private GPA access)	Non-TEE-TLP (Shared GPA access)
TEE-MMIO	TEE-MMIO (V)	TEE-MMIO (X) *
Non-TEE-MMIO	Non-TEE-MMIO (V) **	Non-TEE-MMIO (V)

\* This is rejected by DSM. See Table 3-4.

\*\* A TVM may choose to use TEE-TLP for Non-TEE-MMIO access. In this case, the TVM owner should be aware that **the private GPA usage with TEE-TLP cannot guarantee the data security because DSM has no responsibility to protect the data in Non-TEE-MMIO**. The TVM owner shall still treat the Non-TEE-MMIO data from the device as untrusted data and shall not store any confidential data to Non-TEE-MMIO region, even if the Non-TEE-MMIO is mapped to private GPA.

NOTE:

1. The device shall expose all TEE MMIO resources using **64-bit BARs**.
2. The device shall enforce that IDE Selective Stream bound to TDI and TDI TEE MMIO ranges are configured to **MMIO-H**. Note that this restriction does not apply to Non-TEE-MMIO.
3. The device shall treat any TEE-MMIO BAR re-programming as an error when the TDI is in CONFIG\_LOCKED state.

Figure A-1 shows the trusted MMIO flow initiated from host TEE to the device. Please refer to [Intel TDX Connect] for more details.

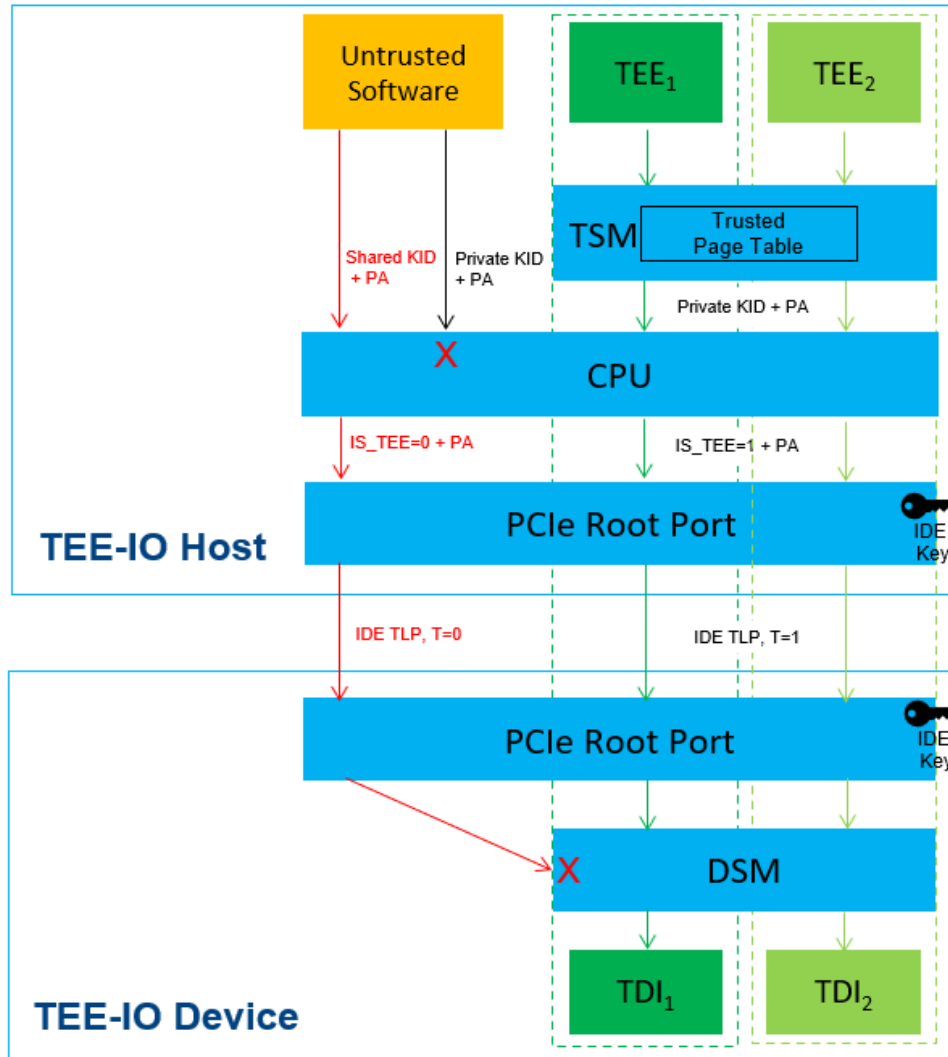


Figure A-1: Trusted MMIO Flow

DMA Access Control

The memory resource in TDX Connect host is defined as following:

- **Private Memory:** The physical memory range whose GPA.S = 0 with private HKID. It is TEE memory in TDX Connect host.
- **Shared Memory:** All other memory which is not private memory. It is Non-TEE memory in TDX Connect host.

Note: The device does not know if the host memory is private memory or shared memory, according to [PCIe TDISP 1.0].

Role and Responsibility:

- **TVM is the policy maker.** TVM may choose to change the shared memory region. TVM shall clear the secret in the private memory before transiting it to shared memory. TVM shall accept the memory configuration and IOMMU configuration from VMM.
- **Intel TSM is the policy enforcer.** Intel TSM provides API (TDG.DMAR.ACCEPT) for the TVM to let TVM accept the Intel IOMMU (VT-d) configuration. While a TDI is in RUN state, **the TSM must pin the DMA page and ensure the DMA cannot be blocked by the VMM.** Otherwise, the VMM may block the DMA page from being accessed by the TDI. The DSM or TDI cannot know that a DMA Write TPL is dropped, because DMA Write TPL is PR without CPL.

The following table A-8 shows the DMA Rules for TDX Connect host as Completer.

**Table A-8: DMA Rule for TDX Connect host as Completer**

Access Control	TEE-TLP	Non-TEE-TLP
Private Memory	Private Memory (V)	Private Memory (X) *
Shared Memory	Shared Memory (V)	Shared Memory (V)

\* It is rejected by the TDX Connect host SOC. TVM defines private memory range. Intel TSM enforces the configuration in TDX Connect host SOC with Intel IOMMU.

Note: Intel TDX Connect SOC supports “P2P via Root Complex”. In this case, Private Memory means the TEE-MMIO on the target device. Shared Memory means the Non-TEE-MMIO on the target device.

Figure A-2 shows the trusted DMA flow initiated from the device to the host. Please refer to [Intel TDX Connect] for more details.

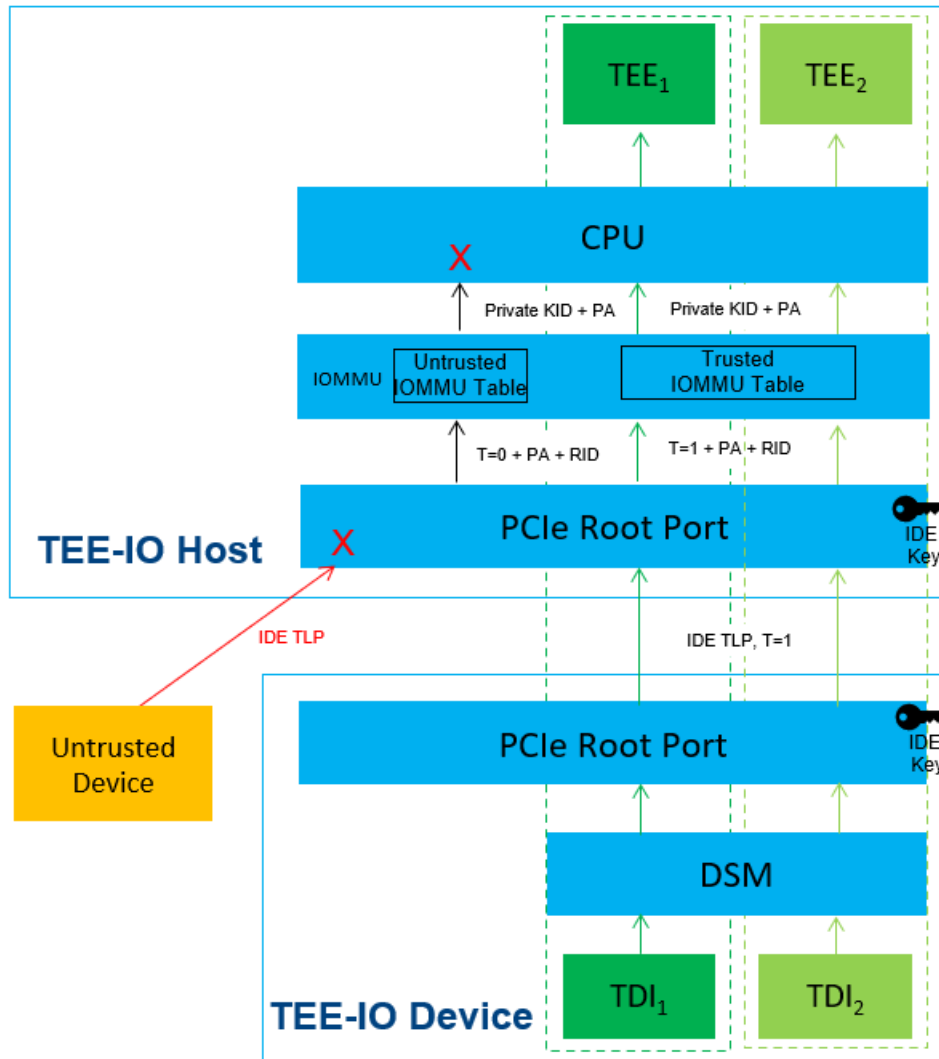


Figure A-2: Trusted DMA Flow

### MSI-X

The TDX Connect host does not support Trusted MSI.

The Intel TSM will not allow setting the *LOCK\_MSIX* flag in the TDISP Lock Interface Request so the device will not use MSI/X requests for TEE-IO transactions with the TDX Connect host.

The device shall clear *MSI\_X\_MESSAGE\_CONTROL* field in TDI Report Structure, otherwise the Intel TSM will reject the Report and will abort transitioning the TDI to the Run state.

### **ATS**

The TDX Connect host does not support ATS.

The device must not use ATS requests for TEE-IO transactions. The device shall not generate Translation Request, Translated Request, or Page Requests on TDIs. During TDI configuration the device could either disable ATS for the specific TDI or it can be turned off globally for the entire device.

The device shall indicate ATS is not supported and enabled in the TDI Report Structure.

### **PRS**

The TDX Connect host does not support PRS.

The device must not use PRS requests for TEE-IO transactions. During TDI configuration it is recommended to disable PRS.

The device shall indicate PRS is not supported and enabled in the TDI Report Structure.

### **Peer to Peer (P2P)**

#### *Direct P2P*

The TDX Connect host SOC does not support Direct P2P or P2P with translated address, because ATS is not supported. In addition, the host SOC does not support setting up the P2P IDE Selective Streams required for Direct P2P. The Intel TSM will prevent the configuration of direct P2P connections.

#### *P2P via Root Complex*

TDX Connect host SOC supports P2P via Root Complex. All P2P messages must be relayed thru the TDX Connect host's Root Complex.

The device may send and handle trusted TLPs only via the Root-Complex IOMMU in TDX Connect CPU host.

### **PASID**

The TDX Connect host does not support PASID. The Intel TSM will not enable PASID on TDIs

The device shall indicate that PASID is not supported and enabled in the TDI Report Structure.

### **Shared Virtual Memory (SVM)**

The TDX Connect host does not support Shared Virtual Memory (SVM), also known as Shared Virtual Addressing. SVM requires PRS and PASID which are not useable with the TDX Connect host.

The TDX Connect host's Trusted IOMMU engine, Intel VT-d, does not support first stage translation which is required for SVM.

## Appendix A: Intel® TDX Connect Interoperability

The Intel TSM will not configure SVM for TDX Connect usage.

### **LNR**

The TDX Connect host SOC does not support Lightweight Notification (LN) Protocol. LN has been deprecated from [PCIe 6.0]

The device should not implement the deprecated Lightweight Notification (LN) protocol, i.e., issue LN Reads, LN Completions, and LN Writes or generate LN metadata.

The device shall clear LNR\_CONTROL in the TDI Report Structure to indicate it is not supported.

### **TPH**

The TDX Connect host SOC does not support TPH due to not supporting MSI-X table locking, which is a requirement for TPH support in [PCIe TDISP 1.0].

The Intel TSM will not configure *TPH Requester Enable* on the TDI

The device shall clear TPH\_CONTROL in the TDI Report Structure.



## Appendix B: Secure Device Interface Lifecycle Example

This section provides more detail SPDM management, IDE stream management and TDI lifecycle management flow as example.

### B.1. SPDM Management

#### B.1.1. SPDM Session Setup

In order to use the device, VMM needs to invoke TSM to establish SPDM session with the device.

Table B-1: SPDM Session Setup Example

Step	Action	Description
1	VMM checks device capability.	VMM reads device PCI DOE capability and IDE capability. VMM uses <b>DOE message DISCOVERY</b> to get the DOE capability.
2	VMM invokes TSM to establish SPDM connection.	TSM sends <b>SPDM messages</b> including <i>GET_VERSION</i> , <i>GET_CAPABILITIES</i> , <i>NEGOTIATE_ALGORITHMS</i> , and processes the responses.
3	VMM invokes TSM to establish SPDM session.	TSM sends <b>SPDM messages</b> including <i>GET_CERTIFICATE(slot=0)</i> , <i>KEY_EXCHANGE</i> , <i>FINISH</i> , and processes the responses.
4	VMM invokes TSM to collect device certificates and measurements from device.	TSM sends <b>SPDM messages</b> including <i>GET_CERTIFICATE(slot=1~7)</i> , <i>GET_MEASUREMENT</i> in SPDM session and processes the responses.

#### B.1.2. SPDM Session Termination

If all TDIs in a device are stopped, the VMM can terminate the SPDM session with the device.

Table B-2: SPDM Session Termination Example

Step	Action	Description
1	VMM may ask TSM to terminate the SPDM session.	VMM invokes TSM to send <b>SPDM message - END_SESSION</b> and processes the responses.



### B.1.3. SPDM Session Heartbeat

The VMM should calculate the time for the SPDM session and invoke the TSM to send heartbeat before the session timeout.

Table B-3: SPDM Session Heartbeat Example

Step	Action	Description
1	VMM may ask TSM to maintain the SPDM session at runtime, with heartbeat.	TSM sends <b>SPDM messages</b> such as <i>HEARTBEAT</i> to keep SPDM session alive.

### B.1.4. SPDM Session Key Update

The VMM should invoke TSM to perform Key Update before the session sequence number overflows. The TSM shall detect the sequence number overflow and stop the session immediately if that happens in either direction.

Table B-4: SPDM Session Key Update Example

Step	Action	Description
1	VMM may ask TSM to maintain the SPDM session at runtime, with key update.	TSM sends <b>SPDM messages</b> such as <i>KEY_UPDATE</i> to keep SPDM session alive.

### B.1.5. Device Information Recollection

If the device supports runtime update, the TVM may want to recollect the device certificates and measurements information to know if the device is kept up to date.

Table B-5: SPDM Device Information Recollection Example

Step	Action	Description
1	TVM may ask TSM to recollect the device certificates and measurements	TSM sends <b>SPDM message</b> – <i>GET_CERTIFICATE</i> and <i>GET_MEASUREMENT</i> again in the SPDM session and returns information to TVM. TVM can verify the certificates and measurements again according to the device attestation policy.



## B.2. IDE Stream Management

### B.2.1. IDE Stream Setup

The host software may use the following sequence in Table 1-7 to start IDE stream with the device. For host SOC, we use Intel Root Complex as an example [Intel RC IDE].

If a step includes the actions for both the host SOC and the device, then the actions for the host SOC and the device can be in any order.

Note: This is not the only sequence to start IDE stream. This is one possible sequence to support both link IDE stream and selective IDE stream.

**Table B-6: IDE Stream Setup Example**

Step	To host SOC	To device	Comment
1	Detect the IDE capability.	Detect the IDE capability.	Stop if no capability.
2	Ensure IDE is disabled.	Ensure IDE is disabled.	This is to avoid confusion with IDE Stream Key Refresh.
3		Enable device's interrupt, if needed.	It is for DOE message.
4		Establish SPDM session	It is for IDE_KM messages and TDISP messages later.
5	Program IDE key to SOC key programming registers.	Send IDE_KM message – KEY_PROG and process the KP_ACK response.	The KEY_PROG is for each IDE sub-stream in (PR, NPR, CPL) and direction in (Rx, Tx).
Now the IDE stream is in ready sub-state.			
6	Program IDE key set for SOC in Rx and Tx.	Send IDE_KM message – K_SET_GO and process the K_GOSTOP_ACK response in Rx and Tx.	The K_SET_GO is for each IDE sub-stream in (PR, NPR, CPL) and direction in (Rx, Tx).
7		Do necessary action to ensure the device will not initiate IDE TLPs before the host SOC is ready. *	For example: Disable device's capability to generate asynchronous TLP. E.g. BusMasterEnable (BME), InterruptEnable.
8		Set IDE Stream Enable bit for the device and check Status bit.	The port must return the configuration completion as a Non-IDE TLP, then trigger the start of IDE. [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE_KM) Page 24.



Now the device port is ready to receive IDE TLP. The device port shall continue to accept Non-IDE TLPs until it receives an IDE TLP. [PCIe IDE 1.0] 6.99.4 IDE TLPs Page 36.			
9	Set IDE Stream Enable bit for SOC		
Now both sides can use IDE-TLP to transmit and receive. The IDE stream is in Secure state.			
10		Re-enable device capability to initiate IDE TLPs, if it is disabled.	For example, BME, InterruptEnable.

**NOTE:**

1. In link IDE stream use case, step 8 must be prior to step 9. If we put step 9 before step 8, the host SOC would send an IDE-TLP for PCI CFG cycle to the device to set IDE Stream Enable bit before the device is ready to receive an IDE-TLP. The consequence is that the device will reject this IDE-TLP, and there is no way to enable the link IDE stream. This is not applicable to selective IDE if the Selective IDE for Configuration Requests Enable bit is cleared.

2. Step 7 is introduced according to [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE\_KM) Page 24. *“System software must ensure that the Partner Ports initiate IDE TLPs sequenced appropriately so that a Port will not receive an IDE TLP before the Enable bit has been set.”* Otherwise, the device might send an IDE-TLP for the interrupt to the host SOC before the host SOC is ready to receive an IDE-TLP. The consequence is that the host SOC will reject this IDE-TLP and lose the interrupt.

Software shall not enable device Tx direction (in step 6) after step 9, because the device shall return the IDE-TLP Completion for the IDE-TLP Non-Posted Request from the host. Otherwise, it will cause IDE Check Failed error. [PCIe IDE 1.0] 6.99.8 Other IDE Rules Page 46.

3. After step 8, the device may still receive Non-IDE TLP until the first IDE TLP. However, if the IDE TLP is Non-Posted Request, then the Completion shall use the same Stream ID and same T bit value. [PCIe IDE 1.0] 6.99.8 Other IDE Rules Page 46. If the device returns Completion with IDE-TLP, the host is not ready yet and will reject such TLP.

**B.2.2. IDE Stream Stop**

The host software may use following sequence in Table 1-8 to stop IDE stream with the device.

**Table B-7: IDE Stream Stop Example**

Step	To host SOC	To device	Comment
1		Clear IDE Stream Enable bit for the device and check Status bit.	The port must return the configuration completion as an IDE TLP, then stop IDE. [PCIe IDE 1.0] 6.99.3 IDE



			Key Management (IDE_KM) Page 24.
Now the device shall only use Non-IDE-TLP to transmit or receive TLPs. The IDE stream is in Insecure state. The host SOC still has IDE enabled. If the host uses IDE-TLP to communicate with the device, the device will reject.			
2	Clear IDE Stream Enable bit for SOC.		
Now both sides shall use Non-IDE-TLP to transmit or receive.			
3		(Optionally) Send IDE_KM message – K_SET_STOP and process the K_GOSTOP_ACK response.	The K_SET_STOP is for each IDE sub-stream in (PR, NPR, CPL) and direction in (Rx, Tx).

## NOTE:

1. In link IDE stream use case, step 1 must be prior to step 2. If we put step 2 before step 1, the host SOC would send a Non-IDE-TLP for PCI CFG cycle to the device to clear IDE Stream Enable bit, but the device can only accept an IDE-TLP. The consequence is that the device will reject this Non-IDE-TLP, and there is no way to disable the link IDE stream gracefully.

2. Once the IDE stream is disabled in step 1, the device can only send Non-IDE-TLP. If the device needs to generate an interrupt before step 2, the interrupt will be lost in link IDE stream use case.

### B.2.3. IDE Stream Key Refresh

The host software may use following sequence in Table 1-9 to refresh IDE keys with the device.

**Table B-8: IDE Stream Key Refresh Example**

Step	To host SOC	To device	Comment
1	Ensure IDE is enabled.	Ensure IDE is enabled.	This is to avoid confusion with IDE Stream Start.
2	Program new IDE key to SOC key programming registers with new key set.	Send IDE_KM message – KEY_PROG and process the KP_ACK response with new key set.	The KEY_PROG is for each IDE sub-stream in (PR, NPR, CPL) and direction in (Rx, Tx).
3	Trigger new IDE key set for SOC in Rx direction.	Send IDE_KM message – K_SET_GO and process the K_GOSTOP_ACK response with new key set in Rx direction.	The K_SET_GO is for each IDE sub-stream in (PR, NPR, CPL) and direction in (Rx).



Now SOC and device still use old keys for transmit and receive, because the port shall continue to accept IDE TLPs using the established key set until it receives an IDE TLP using the new key set. [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE_KM) Page 23.			
4	Trigger new IDE key set for SOC in Tx direction.	Send IDE_KM message – K_SET_GO and process the K_GOSTOP_ACK response with new key set in Tx direction.	The K_SET_GO is for each IDE sub-stream in (PR, NPR, CPL) and direction in (Tx).
Now SOC and device will use new keys for transmitting and receiving. The port shall be ready to use the new key set no more than 10ms after the receipt of the K_SET_GO. The old keys shall be invalidated and never be used if the port receives IDE TLP with new key. [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE_KM) Page 23.			

## NOTE:

According to [PCIe IDE 1.0] 6.99.3 IDE Key Management (IDE\_KM) Page 23, “*The agent implementing the Requester role for IDE\_KM must send K\_SET\_GO commands to enable the Receivers at both IDE Partner Ports, and then send K\_SET\_GO commands to enable the Transmitters at both IDE Partner Ports.*”

## B.3. TDI Lifecycle Management

### B.3.1. TDI Assignment

The VMM follows the above software flow to initialize the device resource and assign to TEE. There are two resource configuration models:

- **Static Configuration Model:** VMM allocates resources when the TDI is initialized and hopes the TEE can accept the resource later.
- **Lazy Configuration Model:** VMM does not allocate resources for the TDI until the TEE starts to accept the configuration.

Table B-9: TDI Assignment Example

Step	Action	Description
1	VMM invokes TSM to negotiate with DSM.	TSM sends <b>TDISP messages</b> including <i>TDISP_GET_VERSION</i> , <i>TDISP_GET_CAPABILITIES</i> , and processes the responses.
2	VMM creates device interface context	VMM creates device interface context for the TDI in TDISP CONFIG_UNLOCKED state.



	and configures the TDI.	VMM allocates MMIO resources for the device.  In <b>static configuration model</b> , VMM builds the trusted DMA table between the TDI and the TVM private memory.
3	VMM invokes TSM to move it to TDISP <b>CONFIG_LOCKED</b> state.	TSM sends <b>TDISP message – LOCK_INTERFACE_REQUEST</b> and processes the response.
4	VMM assigns TDI to TVM.	VMM exposes the device interface to a TVM, for example, PCI Express enumeration.  TVM gets the TDI associated device interface handle from the VMM.
5	TVM gets the device certificates and measurements, then TVM verifies the information.	TVM gets the device SPDM certificates and SPDM measurements. TVM device verifier should verify the certificates and the measurements according to the device attestation policy (policy definition is out of scope of this document).  Finally, TVM calls TSM to accept the device.
4	TVM gets device interface report and verifies it.	TVM invokes TSM to send <b>TDISP message – GET_DEVICE_INTERFACE_REPORT</b> , then get the response. TVM verifies the fields in the device interface report, such as if firmware update is permitted, if TDI generates DMA request with or without PASID, etc.
5	TVM accepts the MMIO and DMA mappings in the report.	TVM accepts all device MMIO pages described in all MMIO_RANGES in the report. TVM accepts DMAR information.  In <b>lazy configuration model</b> , VMM will get an exception when TVM performs the accept, then the VMM can add those resources, and they can be accepted by TVM.
6	TVM calls TSM and VMM to move the TDI into TDISP <b>RUN</b> state.	TVM invokes TSM to send and receive <b>TDISP message – START_INTERFACE_REQUEST</b> , then get the response.
7	TVM pins/unpins private GPA pages as part as DMA allocate/free.	TVM may invoke VMM to ask DMA paging, if the paging is not requested before.  Then TVM can accept DMA paging. After this step, the TVM can perform trusted DMA and MMIO access directly to the TDI.



### B.3.2. TDI Teardown

If a TDI or a device is no longer needed, the VMM may need to stop the TDI or stop the device. There are two possible teardown models:

- **TVM initiated teardown model:** A TVM can actively teardown a TDI. For example, after a TVM performs device information recollection, the TVM does not want to trust the TDI anymore because its version is lower than expected. Or a TVM wants to let the orchestrator initiate a device runtime update. In this mode, the TVM needs to notify VMM to stop the TDI.
- **VMM initiated teardown mode:** If TVM does not teardown the TDI, the VMM needs to ensure that the TDI is stopped gracefully after the TVM itself goes through teardown mode.

Table B-10: TDI Teardown Example

Step	Action	Description
1	TVM asks VMM to stop the TDI. (Optional, only in <b>TVM initiated teardown model</b> )	The TVM may notify VMM to stop the TDI.  After the VMM returns the VMCALL, the TVM may check TDI state to see if it is changed to TDISP CONFIG_UNLOCKED.  Following steps are common for both TVM initiated and VMM initiated TDI stop.
2	VMM moves the TDI to <b>CONFIG_UNLOCKED</b> state.	VMM invokes TSM to generate <b>TDISP message – STOP_INTERFACE_REQUEST</b> and process the response.
3	VMM removes DMA pages and MMIO pages.	VMM removes entries in IOMMU's DMA remapping structure.  VMM blocks MMIO pages, invalidates the IOTLB and unmaps MMIO pages from a TVM.

## References

### Standards

[SPDM 1.2] DMTF, DSP0274 - Secure Protocol and Data Model, Version 1.2.1, June 2022, <https://www.dmtf.org/dsp/DSP0274>

[Secured SPDM 1.1] DMTF, DSP0277 - Secured Messages Using SPDM, Version 1.1, May 2022, <https://www.dmtf.org/dsp/DSP0277>

[SPDM WP 1.1] DMTF, DSP2058 - Security Protocol and Data Model (SPDM) Architecture White Paper, Version 1.1, Feb 2022, <https://www.dmtf.org/dsp/DSP2058>

[SPDM Val] DMTF, IS0023 - SPDM Conformance Test Suite Guidance, <https://www.dmtf.org/dsp/DSP-IS0023>

[PCIe 6.0] PCI-SIG, PCI Express Base Specification 6.0.1, Version 1.0, August 2022, <https://members.pcisig.com/wg/PCI-SIG/document/18363>

[PCIe 6.1] PCI-SIG, PCI Express Base Specification 6.1.0, Version 1.0, July 2023, <https://members.pcisig.com/wg/PCI-SIG/document/19849>

[PCIe 6.2] PCI-SIG, PCI Express Base Specification 6.2.0, Version 1.0, January 2024, <https://members.pcisig.com/wg/PCI-SIG/document/20590>

[PCIe DOE 1.0] PCI-SIG, Data Object Exchange (DOE) 1.0 ECN, March 2020, <https://members.pcisig.com/wg/PCI-SIG/document/14143> or [PCIe 6.0] Section 6.30 – Data Object Exchange (DOE)

[PCIe DOE 1.1] PCI-SIG, Data Object Exchange (DOE) 1.1 ECN, September 2022, <https://members.pcisig.com/wg/PCI-SIG/document/18483> or [PCIe 6.1] Section 6.30 – Data Object Exchange (DOE)

[PCIe CMA 1.0] PCI-SIG, Component Measurement and Authentication (CMA) 1.0 ECN, April 2020, <https://members.pcisig.com/wg/PCI-SIG/document/14236> or [PCIe 6.0] Section 6.31 – Component Measurement and Authentication (CMA)

[PCIe CMA 1.0 Revised] PCI-SIG, Component Measurement and Authentication (CMA) 1.0 Revised ECN, October 2023, <https://members.pcisig.com/wg/PCI-SIG/document/20110> or [PCIe 6.2] Section 6.31 – Component Measurement and Authentication (CMA)

[PCIe IDE 1.0] PCI-SIG, Integrity, and Data Encryption (IDE) 1.0.A ECN, October 2021, <https://members.pcisig.com/wg/PCI-SIG/document/16599> or [PCIe 6.0] Section 6.33 – Integrity and Data Encryption (IDE)

## References

- [PCIe TDISP 1.0] PCI-SIG, TEE Device Interface Security Protocol (TDISP) 1.0 ECN, July 2022, <https://members.pcisig.com/wg/PCI-SIG/document/18255> or [PCIe 6.1] Chapter 11 – TEE Device Interface Security Protocol (TDISP)
- [CXL 3.0] CXL Consortium, Compute Express Link (CXL) Specification 3.0, Version 1.0, August 2022, <https://www.computeexpresslink.org/download-the-specification>
- [DICE HW] TCG, Hardware Requirements for a Device Identifier Composition Engine, March, 2018, <https://trustedcomputinggroup.org/resource/hardware-requirements-for-a-device-identifier-composition-engine/>
- [DICE Attestation] TCG, DICE Attestation Architecture, March 2021, <https://trustedcomputinggroup.org/resource/dice-attestation-architecture/>
- [DICE Layering] TCG, DICE Layering Architecture, July 2020, <https://trustedcomputinggroup.org/resource/dice-layering-architecture/>
- [DICE Endorsement] TCG, DICE Endorsement Architecture for Device, May 2022, <https://trustedcomputinggroup.org/resource/dice-endorsement-architecture-for-devices-v1-0-r0-38/>
- [DICE Evidence Binding SPDM] TCG DICE Concise Evidence Binding for SPDM, 2025, [https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Concise-Evidence-Binding-for-SPDM-Version-1.1-RC1\\_10April25.pdf](https://trustedcomputinggroup.org/wp-content/uploads/TCG-DICE-Concise-Evidence-Binding-for-SPDM-Version-1.1-RC1_10April25.pdf)
- [TCG Comp RIM IM] TCG draft, TCG Component Reference Integrity Manifest Information Model, 2024, [https://trustedcomputinggroup.org/wp-content/uploads/TCG-Component-Reference-Integrity-Manifest-Information-Model-Version-1.0-Revision-41\\_11December24.pdf](https://trustedcomputinggroup.org/wp-content/uploads/TCG-Component-Reference-Integrity-Manifest-Information-Model-Version-1.0-Revision-41_11December24.pdf)
- [TCG Comp RIM Binding SWID] TCG draft, TCG Component RIM Binding for SWID/CoSWID, 2025, [https://trustedcomputinggroup.org/wp-content/uploads/TCG-Component-RIM-Binding-for-SWID-and-CoSWID-Version-1.0-RC-2\\_16April25.pdf](https://trustedcomputinggroup.org/wp-content/uploads/TCG-Component-RIM-Binding-for-SWID-and-CoSWID-Version-1.0-RC-2_16April25.pdf)
- [CyRes] Cyber Resilient Module and Building Block Requirements, June 2022, <https://trustedcomputinggroup.org/resource/cyber-resilient-module-and-building-block-requirements/>
- [SP 800-193] NIST, Platform Firmware Resiliency Guidelines, May 2018, <https://csrc.nist.gov/publications/detail/sp/800-193/final>
- [IETF-RATS] IETF, RFC 9334: Remote Attestation Procedures Architecture, <https://datatracker.ietf.org/doc/rfc9334/>
- [IETF-ATTS] IETF draft, Reference Interaction Models for Remote Attestation Procedures, <https://datatracker.ietf.org/doc/draft-ietf-rats-reference-interaction-models/>

## References

[IETF-CORIM] IETF draft, Concise Reference Integrity Manifest (CoRIM),  
<https://datatracker.ietf.org/doc/draft-ietf-rats-corim/>

[IETF-COSWID] IETF, RFC 9393, Concise Software Identification Tags (CoSWID),  
<https://datatracker.ietf.org/doc/rfc9393/>

[S-IOV] OCP, Scalable I/O Virtualization Revision 1.0, Version 1.2, Feb. 2022,  
<https://www.opencompute.org/documents/ocp-scalable-io-virtualization-technical-specification-revision-1-v1-2-pdf>

[Intel S-IOV] Intel Scalable I/O Virtualization Revision 1.1, September 2022,  
<https://www.intel.com/content/www/us/en/develop/download/intel-scalable-io-virtualization-technical-specification.html>

[Intel VT-d] Intel Virtualization Technology for Directed I/O Architecture Specification, Revision 4.0, June 2022, <https://www.intel.com/content/www/us/en/develop/download/intel-virtualization-technology-for-directed-io-architecture-specification.html>

[Intel RC IDE] Intel Root Complex IDE Key Configuration Unit - Software Programming Guide, June 2022, <https://www.intel.com/content/www/us/en/io/pci-express/pci-express-architecture-devnet-resources.html>

[Intel TDX Connect] Intel TDX Connect Architecture Specification, January 2023,  
<https://software.intel.com/content/www/us/en/develop/articles/intel-trust-domain-extensions.html>

[AMD-SEV] AMD, AMD Secure Encrypted Virtualization (SEV), <https://developer.amd.com/sev/>

[ARM-RME] ARM, ARM Realm Management Extension (RME),  
<https://developer.arm.com/documentation/den0129>

[RISCV-APTEE] RISCV, RISCV AP TEE, <https://github.com/riscv-non-isa/riscv-ap-tee>

## Web Resources

[RATLS] Integrating Remote Attestation with Transport Layer Security (RA-TLS),  
<https://github.com/cloud-security-research/sgx-ra-tls>

[OPENENCLAVE] open enclave, <https://github.com/openenclave/openenclave>

[OPENENCLAVE-RATLS] attested TLS,  
[https://github.com/openenclave/openenclave/blob/master/samples/attested\\_tls/AttestedTLS\\_README.md](https://github.com/openenclave/openenclave/blob/master/samples/attested_tls/AttestedTLS_README.md)

[DEV-ATT-TEE] White paper - Device Attestation Model in Confidential Computing Environment, <https://software.intel.com/content/www/us/en/develop/articles/intel-trust-domain-extensions.html>

