SEV-TIO Firmware Interface Specification

Technical Preview

Publication # **58271** Revision: **0.70**
Issue Date: **May, 2023**

Advanced Micro Devices

Chapter 1 Introduction

1.1 Purpose

The purpose of this document is to specify the SEV Trusted I/O (SEV-TIO) extension to the SEV firmware. The SEV-TIO extension provide a mechanism for guests to bind to and use trusted devices within their guest private address space.

1.2 Scope

This document specifies the software interface for functions supported by the AMD Secure Processor (ASP) for SEV-TIO trusted device management. It does not describe the x86 CPU or System-on-Chip (SoC) hardware support for SEV-TIO. While certain sections of this document may describe potential host software usage of the firmware interface, this document is not intended to prescribe any specific use or host software architecture.

This document is a technical preview. Future versions of this document may introduce backward compatibility–breaking changes.

1.3 Intended Audience

The intended audience of this document are the host software developers and security architects. Host software developers supporting SEV-TIO must use the firmware functions described herein for device lifecycle management. Additionally, kernel developers and security architects must use the guest message functions to perform device attestation and system configuration to bring devices into the guest trust boundary.

Chapter 2 Theory of Operation

This interface specification extends the Secure Encrypted Virtualization (SEV) and SEV Secure Nested Paging (SEV-SNP) firmware interface specifications [1] [2] with a new set of host commands and guest request messages that allow a guest to establish trust in a device that supports TEE Device Interface Security Protocol (TDISP) [3] and then interact with the device via private memory. Host software is responsible for assigning device and host resources to the guest. Host software is also responsible for facilitating the flows necessary to supporting guests' assessment and usage of TDISP capable devices. For an overview of the feature, please see the AMD SEV-TIO whitepaper [4].

The AMD Security Processor (ASP) executes the SEV firmware that services all host commands and guest request messages. Through this interface, host software orchestrates TDISP between the SEV firmware, which serves as the TEE Security Manager (TSM), and the Device Security Manager (DSMs) of TDISP capable devices.

The following sections describe the functionality of the SEV-TIO interface provided by the SEV firmware.

2.1 Feature Detection

Software can detect whether the host hardware supports SEV-TIO according to the AMD IOMMU specification [5]. To detect that the currently loaded SEV firmware supports SEV-TIO, software can retrieve bit 1 from Fn80000024_EBX_x00 using the FEATURES_INFO command.

2.2 Platform Initialization

SEV-TIO is initialized by setting the TIO_EN flag to 1 in the SNP_INIT_EX command. When TIO_EN is 0, SNP_INIT_EX initializes the RMP entries for each MMIO pages in the platform to HV-fixed, which prevents host software from assigning MMIO pages to guests in the RMP. When TIO_EN is 1, the command leaves the pages in the Hypervisor page state, which allows host software the ability to assign MMIO pages to guests. SNP_INIT_EX also initializes the SEV-TIO feature in the IOMMUs. Subsequent invocations of SNP_INIT_EX that do not request RMP re-initialization must have the same TIO_EN value as the previous invocation.

2.3 Scatter Lists and Buffers

SEV-TIO uses buffers that may span multiple noncontiguous physical pages. Because the ASP accesses system physical addresses directly without address translation, SEV firmware uses a scatter list to reference noncontiguous physical pages. When host software allocates a buffer for SEV-TIO to use, software constructs a scatter list that points to each physical page of the buffer. Scatter lists are used to reference two new types of host-donated context buffers for devices and their interfaces, as well as communication buffers for transmitting TDISP related data during the invocation of TIO commands.

The logical buffer constructed by scatter lists contains a header that describes the contents of the buffer. The remainder of the buffer is a sequence of data objects. Some data objects are intended to be constructed or parsed by host software, which are specified in Section 4.5. Other data objects contain implementation-specific data intended to be used only by SEV firmware that should be managed by host software as opaque objects. Buffers containing opaque objects are protected from software modification by the enforcement of RMP page states.

Buffers passed as parameters to commands may require resizing to successfully complete the command. Specifically, when SEV firmware is writing to a buffer, it may determine that the buffer is too small to complete the command. In this case, the firmware writes into the header of the buffer to indicate the minimum buffer size and returns a status code that indicates to host software that the buffer must be expanded in size. The status code returned is command specific.

2.4 Device Context Buffer

SEV firmware tracks the state of a device specific to SEV-TIO in a device context buffer. The host creates a device context buffer for each device by allocating the buffer, transitioning each page of the buffer to the Firmware page state, and then invoking the TIO_DEV_CREATE command. TIO_DEV_CREATE initializes the buffer and transitions the page to the Device-Context page state.

Device context buffers are fixed in size. Host software must allocate a buffer at least as large as the DEVCTX_SIZE field returned by the TIO_STATUS command. The minimum buffer size for context pages does not change until after one of these commands is invoked: SNP_SHUTDOWN, SNP_SHUTDOWN_EX, or DOWNLOAD_FIRMWARE_EX.

To reclaim a device context buffer, host software can invoke TIO_DEV_RECLAIM. Host software must invoke TIO_TDI_RECLAIM on all TDI context buffers of the device before reclaiming a device context buffer. TDI context buffers are described further in Section 2.6.

2.5 Device Connection

To use a TDISP capable device with SEV-TIO, host software must first arrange for the SEV firmware to establish a connection with the device by invoking the TIO_DEV_CONNECT command. The TIO_DEV_CONNECT command performs the following:

- Establishes a secure SPDM session using Secured Messages for SPDM.
- Constructs IDE selective streams between the root complex and the device.
- Checks the TDISP capabilities of the device.

SPDM is a request-response protocol in which the SEV firmware sends request messages to the device, and the device returns response messages back to the SEV firmware. The SPDM messages are cryptographically protected and transported between the SEV firmware and the device via host software. For details on how the SEV firmware sends and receives SPDM messages through host software, see Chapter 5.

2.6 TDI Context Buffer

Each TDISP-capable device provides one or more TDIs that can be securely bound to SNP active guests. The SEV firmware tracks the state of a TDI specific to SEV-TIO with a TDI context buffer. The TDI context buffer size must meet the minimum size requirement reported in the TIO_STATUS command. The host creates a TDI context buffer by allocating the buffer, transitioning each page of the buffer to the Firmware page state, and then invoking the TIO_TDI_CREATE command. TIO_TDI_CREATE initializes the buffer and transitions the page to the TDI-Context page state.

TDI context buffers are fixed in size. Host software must allocate a buffer at least as large as the TDICTX_SIZE field returned by the TIO_STATUS command. The minimum buffer size for context pages does not change until after one of these commands is invoked: SNP_SHUTDOWN, SNP_SHUTDOWN_EX, or DOWNLOAD_FIRMWARE_EX.

To reclaim a TDI context buffer, host software can invoke TIO_TDI_RECLAIM.

2.7 Device Assignment and Binding

Host software binds a TDI to an SNP active guest by first mapping the MMIO ranges of the TDI to the private address space of the guest and transitioning the pages of the ranges to the Pre-Guest page state. Then host software invokes TIO_TDI_BIND.

TIO_TDI_BIND sends request messages to the device to transition the TDI to the CONFIG_LOCKED state and then to the RUN state. Host hardware restricts DMA from the TDI into guest private memory until the guest sends the TIO_MSG_SDTE_WRITE_REQ guest request message to grant the TDI access. The guest also cannot access MMIO ranges of the TDI until the guest sets the RMP.Validated bits of the pages in the ranges using the TIO_MSG_MMIO_VALIDATE_REQ guest request message.

To unbind the TDI from a guest and return the TDI to the CONFIG_UNLOCKED state, host software can invoke the TIO_TDI_UNBIND command.

2.8 Device Attestation

Before the guest enables DMA and MMIO for a TDI, the guest can retrieve attestation information about the device. The attestation objects available for the guest are the device certificate chain, the device attestation report, and the interface report. The policy used to validate these objects is beyond the scope of this specification.

The guest relies on host software to transfer the attestation objects. Host software can retrieve the certificate chain, device attestation report, and interface report using the TIO_DEV_CERTIFICATES, TIO_DEV_MEASUREMENTS, and TIO_TDI_REPORT commands, respectively. Additionally, the TIO_DEV_CONNECT command outputs the certificate chain, and the TIO_TDI_BIND command outputs the interface report to host software.

When the guest receives the attestation objects from host software, it can validate that the attestation objects were not tampered with by retrieving their cryptographic digests from the SEV firmware using the TIO_TDI_INFO_REQ guest request message. This message gives the guest the digest of the attestation objects last retrieved by the SEV firmware.

The guest can ensure that the attestation report is fresh by checking that the MEAS_FRESH flag in the TIO_MSG_TDI_INFO_RSP guest message is 1. MEAS_FRESH indicates that the attestation report was retrieved after the TDI was locked. This may be important to guests to ensure that host software has not made security changes to the device that cause the device attestation report to become stale.

2.9 MMIO Validation

The guest can discover the location of the MMIO ranges of the TDI through existing guest PCIe enumeration. For instance, the guest may find the location of MMIO ranges by reading the Base Address Registers (BARs) out of the PCIe configuration space of the TDI. Because guest configuration space would be emulated by host software, the guest can validate the location and attributes of the MMIO ranges by examining the interface report.

The guest can examine the interface report to determine whether the MMIO range attributes agree with the security policy of the guest. For instance, the guest might wish to ensure that IS_NON_TEE_MEM is 0 for sensitive MMIO ranges that must not be accessible to host software. The guest can also retrieve and update the MMIO range attributes with the TIO_MSG_MMIO_CONFIG_REQ guest message as desired.

After the guest has agreed with the location and security attributes of the MMIO range, the guest can send the TIO_MSG_MMIO_VALIDATE_REQ message. The guest provides the RangeID, base guest physical address, and length of the MMIO range in the message. The firmware checks that the MMIO range is correctly mapped into the guest address space and then sets the Valdiated bit. The guest can also clear the Validated bit for the MMIO range using this guest message.

Note that the guest need not read the MMIO range base addresses in the interface report. The base addresses in the report are derived from system physical addresses that provide no useful information to the guest. Because system physical address space layout is often considered too sensitive to share with guests, the SEV firmware zeroes the base addresses in the report.

Note that the PVALIDATE instruction triggers a #GP exception when used for MMIO ranges.

2.10 Enabling Direct Memory Access

The IOMMU rejects all access attempts of a device to guest private memory until the guest sends the TIO_MSG_SDTE_WRITE_REQ guest request message. This message sets the Secure Device Table Entry (SDTE) for the TDI. To grant the device access to guest private memory, the guest can set the V bit of the SDTE to 1.

The SDTE contains the VMPL to assign to the TDI. On memory access by the TDI, the IOMMU uses the VMPL field of the SDTE for the RMP access check during address translation.

The guest can set the Virtual Top of Memory (vTOM) for the TDI in the SDTE. When vTOM is enabled, the IOMMU determines the C-bit of the TDI access using the provided vTOM address in the SDTE. All data accesses below vTOM are accessed with an effective C-bit of 1, and all addresses at or above vTOM are accessed with an effective C-bit of 0.

***Technical Preview Note**: This revision of the technical preview supports binding devices with guest paging mode disabled. In future revisions, guest paging support will be added, which will expand the definition of the SDTE will be expanded with further fields associated with guest paging.*

2.11 Error Conditions

Each TDI of a TDISP capable device may enter the TDISP ERROR state. When a TDI is in the ERROR state, a TDISP capable device is expected to prevent further access to guest private memory via that TDI. The TIO_TDI_INFO command can be used to retrieve the current state of the TDI and check whether the TDI has entered the ERROR state.

Host software can reclaim a TDI that has transitioned to the ERROR state by invoking TIO_TDI_UNBIND. This transitions the TDI back to the CONFIG_UNLOCKED state. Note that unbinding a TDI from a guest destroys any guest context or data within the TDI.

If a bound TDI sends a request to the root complex, and the IOMMU detects a fault caused by host configuration, the root complex fences the ASID from all further I/O to or from that guest. A host fault is either a host page table fault or an RMP check violation. ASID fencing means that the IOMMU blocks all further I/O from the root complex to the guest that the TDI was bound, and the root complex blocks all MMIO accesses by the guest. When a guest writes to MMIO, the write is silently dropped. When a guest reads from MMIO, the guest reads 1s.

Host software can clear the ASID fencing by unbinding all TDIs on the root complex from the guest. Then, host software invokes the TIO_ASID_FENCE_CLEAR command. The guest cannot interact with the TDIs until the host invokes TIO_TDI_BIND again and the guest re-attests the TDI. The SNP_DECOMMSSION command also clears the ASID fence.

2.12 Device Lifecycle

The commands and guest request messages described in the previous sections are used together in the lifecycle of a TDISP capable device. Figure 1 illustrates this primary steps of this lifecycle.

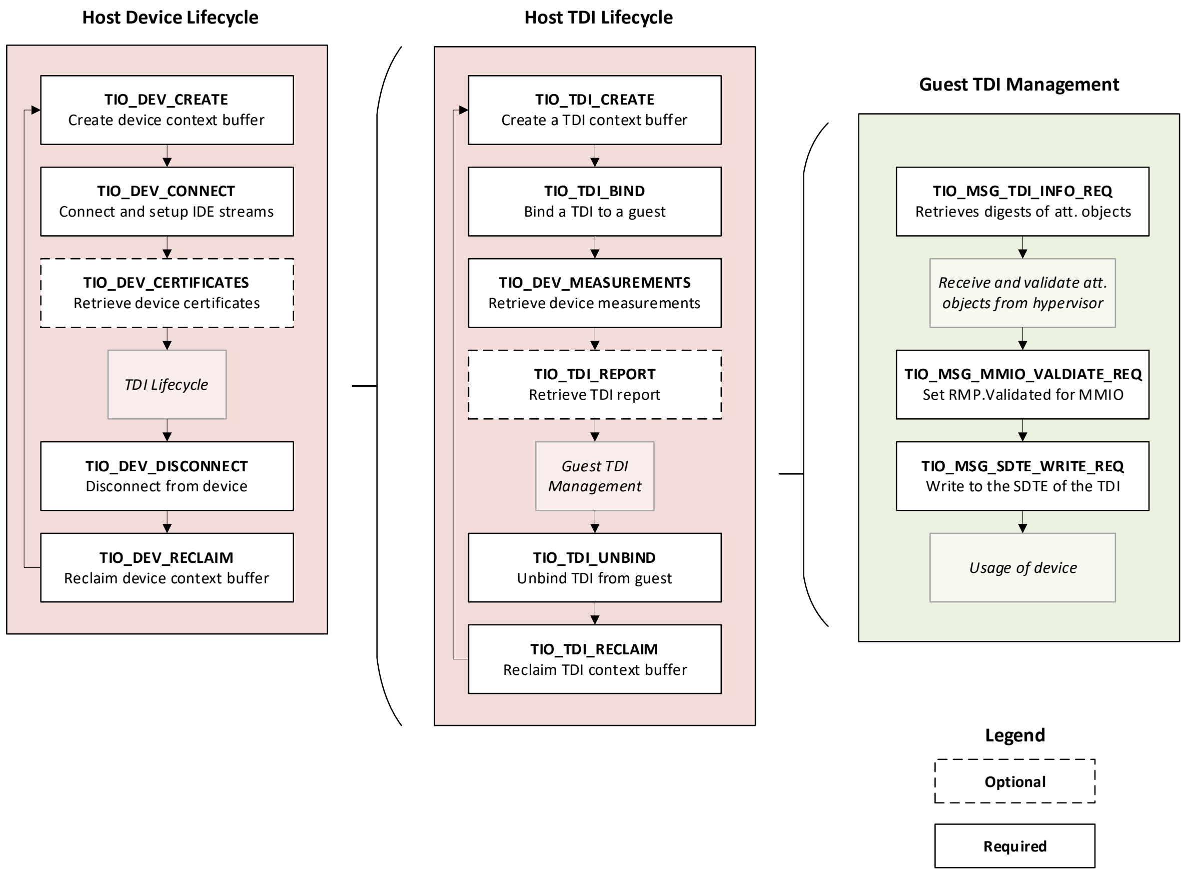

Figure 1: Lifecycle of a TDISP Capable Device

Host software manages the lifecycles of devices and TDIs. As mentioned above, the device certificate chain and the interface reports are output by the TIO_DEV_CONNECT and TIO_TDI_BIND commands respectively, but host software can optionally retrieve them with TIO_DEV_CERTIFICATES and TIO_TDI_REPORT. The TIO_DEV_MEASUREMENTS command should be invoked after TIO_TDI_BIND to retrieve a fresh device attestation report for the guest to consume. Otherwise, the report may be stale, which is reported to the guest via TIO_MSG_TDI_INFO_REQ.

All irrecoverable errors can be cleared and reset by completing the lifecycle flow. For instance, if a guest shuts down unexpectedly, host software can recover the resources associated with the TDIs bound to the guest by invoking the TIO_TDI_UNBIND command. Similarly, device errors like reset, power failure, or physical removal can be handled by invoking TIO_DEV_DISCONNECT. Both commands succeed even if the device is unresponsive.

Chapter 3 Mailbox Protocol

This specification extends the mailbox protocol with the SEV firmware defined in SEV ABI [1] and in SEV-SNP ABI [2].

3.1 Command Identifiers

The command identifiers for SEV-TIO related commands are specified in Table 1.

Table 1. Command Identifiers

Command	ID	Description
TIO_STATUS	D0h	Retrieve status of the SEV-TIO feature.
TIO_DEV_CREATE	D1h	Create a device context buffer.
TIO_DEV_RECLAIM	D2h	Reclaim a device context buffer.
TIO_DEV_CONNECT	D3h	Connect the SEV firmware to a device.
TIO_DEV_DISCONNECT	D4h	Disconnect the SEV firmware from a device.
TIO_DEV_STATUS	D5h	Retrieve status of a device.
TIO_DEV_MEASUREMENTS	D7h	Retrieve the SPDM measurements from a device.
TIO_DEV_CERTIFICATES	D8h	Retrieve the SPDM certificate chain from a device.
TIO_TDI_CREATE	DAh	Create a TDI context page.
TIO_TDI_RECLAIM	DBh	Reclaim a TDI context page.
TIO_TDI_BIND	DCh	Bind a TDI to a guest.
TIO_TD_UNBIND	DDh	Unbind a TDI from a guest.
TIO_TDI_REPORT	DEh	Retrieve the interface report of a TDI.
TIO_TDI_STATUS	DFh	Retrieve status about the TDI.
TIO_GUEST_REQUEST	E0h	Send guest request messages capable of sending SPDM messages.
TIO_ASID_FENCE_CLEAR	E1h	Clear the ASID fencing of a root complex.
TIO_ASID_FENCE_STATUS	E2h	Retrieve the current ASID fencing status for a root port.
TIO_TDI_INFO	E3h	Retrieve TDISP related information about the TDI.

3.2 Status Codes

The status codes for SEV-TIO related commands are specified in Table 2.

Table 2. Status Codes

Status	Code	Description
SPDM_REQ_LENGTH	28h	Indicates that the SPDM request buffer is not large enough.
SPDM_SCRATCH_LENGTH	29h	Indicates that the SPDM scratch buffer is not large enough.
SPDM_OUT_LENGTH	2Ah	Indicates that the output buffer is not large enough.
INCORRECT_BUFFER_LENGTH	2Bh	Indicates that the buffer provided is not large enough.
INVALID_CONTEXT	2Ch	Indicates that the provided context buffer is invalid.
RECLAIM_REQUIRED	2Dh	Indicates that some resources need to be reclaimed before invoking this command.
IN_USE	2Eh	Indicates that the resource is currently in use.

Chapter 4 Scatter Lists and Buffers

To represent a logical buffer across noncontiguous system physical pages, the interface requires software to pass a scatter list. This chapter describes scatter lists and the logical buffers they reference.

4.1 Scatter List Pointer

A Scatter Tree Address (SLA) is an 4 KB aligned system physical address (SPA) that uses the lower 12 bits of the address to store metadata about the page pointed to by the address. Table 3 describes the layout of an SLA.

Table 3. Layout of the Scatter List Address (SLA)

Byte Offset	Bits	Name	Description
0h	63:52	–	Reserved. Must be 0.
	51:12	SPA	Bit[51:12] of a system physical address.
	11:2	–	Reserved. Must be 0.
	1	PAGE_SIZE	0: The page pointed to by this SLA is 4 KB. 1: The page pointed to by this SLA is 2 MB.
	0	PAGE_TYPE	0: The page pointed to by this SLA is a data page. 1: The page pointed to by this SLA is a scatter tree node page.

The full system physical address encoded by the SLA is constructed by setting bits [63:52] and [11:0] to zero and setting bits [51:12] to the value in the SPA field of the SLA. That is, one can calculate the system physical address by masking the SLA with 000FFFFF_FFFFF000h.

If PAGE_TYPE is 1, then PAGE_SIZE must be 0.

4.2 Scatter List

If the PAGE_TYPE field of an SLA is 1, then the SLA points to a scatter list page, which is a page of memory formatted as specified in Table 4.

Table 4. Layout of a Scatter Tree Node

Byte Offset	Bits	Name	Description
0h–FFFh		SLA_ARRAY	Array of 512 SLAs.

If the SPA field of an SLA is all 1s, then the entry is invalid. The firmware reads each SLA_ARRAY entry until it encounters and invalid entry or it encounters the end of the page.

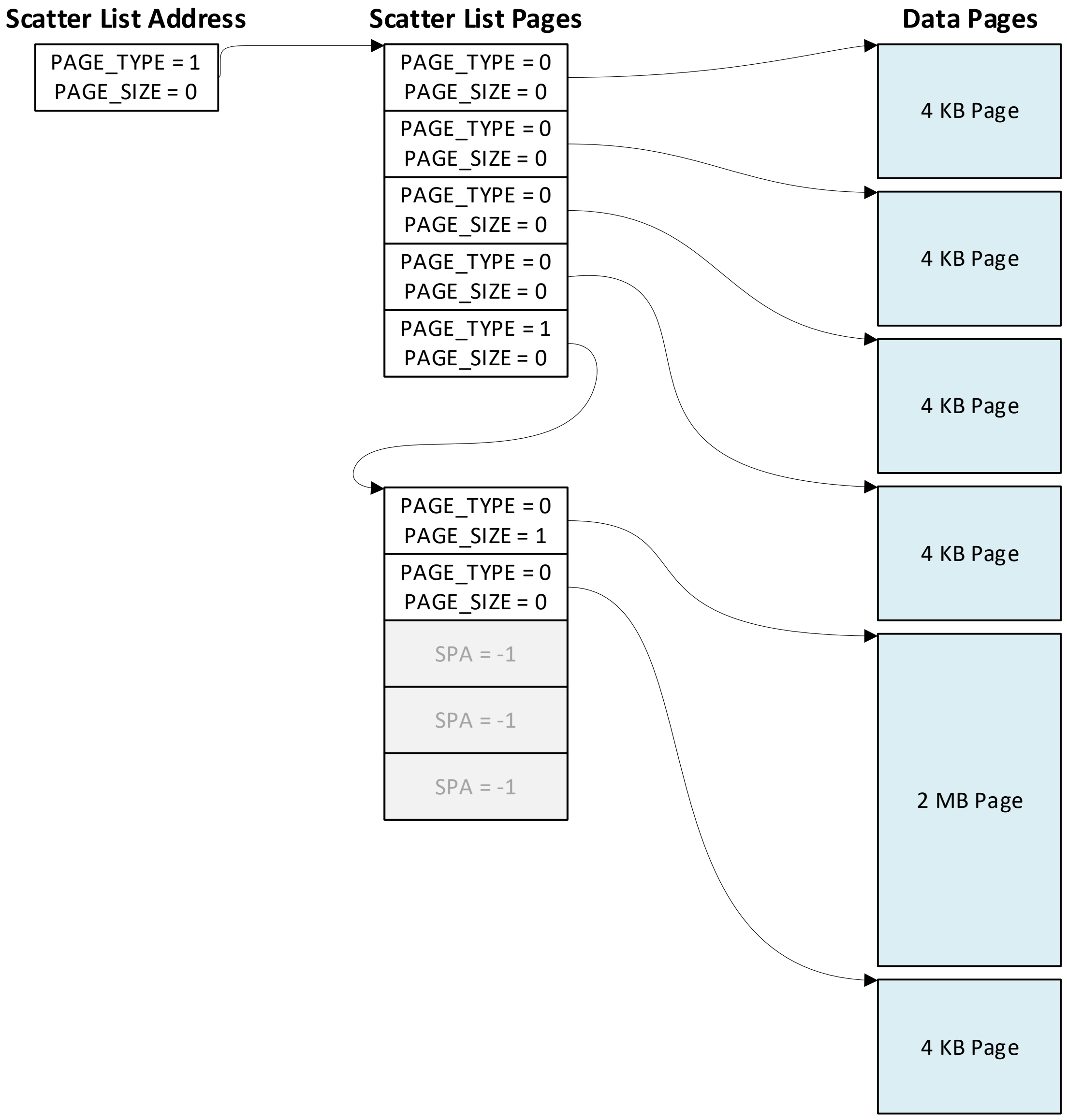

Figure 2: Scatter List and Its Formation of a Buffer

The first 511 entries must have PAGE_TYPE cleared to 0 indicating that the SLA points to a data page. The last entry may have PAGE_TYPE cleared to 0 or set to 1. If PAGE_TYPE is 1, the last entry points to a subsequent scatter list extends the buffer further. The firmware supports up to 512 total chained scatter lists unless otherwise stated.

For example, in Figure 2, a scatter list address points to the first scatter list page. Because the scatter list page is full of valid scatter list pointers, the last entry points to a second scatter list page. The resulting logical buffer is made up of noncontiguous pages of both 4 KB and 2 MB sizes.

4.3 Page States

Because SEV firmware reads only from the scatter list pages, the scatter list pages may be in any page state. The data pages pointed to by the scatter list that form the logical buffer may be required to be in a specific RMP page state depending on the command accessing the buffer.

For instance, the scatter list page of a device context buffer may be in the Hypervisor page state, but each of the data pages that form the device context buffer must be in the Device-Context page state.

4.4 Buffer Layout

The sequence of data pages pointed to by a scatter list form a logical buffer. Every buffer is formatted with a header followed by one or more data objects. Table 5 specifies the structure of a logical buffer.

Table 5. Layout of the BUFFER Structure

Byte Offset	Bits	Name	Description
0h	31:0	BUFFER_CAPACITY	The capacity of the buffer in bytes.
4h	31:0	BUFFER_SIZE	The size of BUFFER_PAYLOAD in bytes. Must be multiple of 32B.
8h	31:0	BUFFER_FLAGS	Bit 0: Indicates that this buffer is encrypted. Bit[31:1]: Reserved.
Ch	31:0	–	Reserved.
10h	127:0	BUFFER_IV	IV used for the encryption of this buffer.
20h	127:0	BUFFER_AUTHTAG	Authentication tag for this buffer.
30h	127:0	–	Reserved.
40h	–	BUFFER_PAYLOAD	BUFFER_SIZE bytes of data.

When the SEV firmware command needs to write to a buffer, but the buffer is not large enough, the SEV firmware writes the desired buffer size into BUFFER_CAPACITY and returns a status code indicating the buffer needs allocation from host software. The status code is determined by the command. See command descriptions for details.

When bit 0 of BUFFER_FLAGS is 1, BUFFER_PAYLOAD is encrypted using a key known only to SEV firmware and the IV stored in BUFFER_IV. When the SEV firmware writes out data to the buffer, it stores the authentication tag into BUFFER_AUTHTAG.

4.5 Data Objects

Each data object stored in a logical buffer has a common header. Table 6 specifies the structure of the data object common header.

Table 6. Layout of the DATA_OBJECT_HEADER Structure

Byte Offset	Bits	Name	Description
0h	31:0	DOBJ_ID	Data object type identifier.
4h	31:0	DOBJ_LENGTH	Length of the data object tin bytes, including this header. Must be a multiple of 32B.
8h	15:0	DOBJ_VERSION	Version of the data object structure. Bit[7:0]: Minor version. Bit[15:8]: Major version.
Ah–Fh		–	Reserved.

The following subsections specify each type of data objects.

4.5.1 SPDM Request Object

A data object containing an SPDM request packet to be sent to a device. See Chapter 5 for usage description of this object.

Table 7. DATA_OBJECT_HEADER for SPDM Request Objects

Byte Offset	Bits	Name	Description
0h	31:0	DOBJ_ID	1h.
4h	31:0	DOBJ_LENGTH	Variable.
8h	15:0	DOBJ_VERSION	0: Minor. 1: Major.
Ah	7:1	–	Reserved
	0	SPDM_TYPE	0: The packet is not secured. 1: The packet is secured.
Bh–Fh		–	Reserved.
10h	–	DOBJ_PAYLOAD	SPDM request packet.

4.5.2 SPDM Response Object

A data object containing an SPDM response packet received from a device to be sent to the SEV firmware. See Chapter 5 for usage description of this object.

Table 8. DATA_OBJECT_HEADER for SPDM Response Objects

Byte Offset	Bits	Name	Description
0h	31:0	DOBJ_ID	2h.
4h	31:0	DOBJ_LENGTH	Variable.
8h	15:0	DOBJ_VERSION	0: Minor. 1: Major.
Ah	7:1	–	Reserved.
	0	SPDM_TYPE	0: The packet is not secured. 1: The packet is secured.
Bh–Fh		–	Reserved.
10h	–	DOBJ_PAYLOAD	SPDM request packet.

4.5.3 SPDM Scratch Object

A data object containing a SPDM context for an SPDM interaction with a device. See Chapter 5 for usage description of this object.

Table 9. DATA_OBJECT_HEADER for SPDM Scratch Objects

Byte Offset	Bits	Name	Description
0h	31:0	DOBJ_ID	3h.
4h	31:0	DOBJ_LENGTH	Variable.
8h	15:0	DOBJ_VERSION	0: Minor. 1: Major.
Ah–Fh		–	Reserved.
10h	15:0	DEVICE_ID	PCIe Routing Identifier of function 0 of the device.
12h	7:0	SEGMENT_ID	PCIe Segment ID.
13h–1Fh		–	Reserved.
20h	–	DOBJ_PAYLOAD	SPDM request packet.

4.5.4 SPDM Certificate Object

A data object containing the certificate chain retrieved from a device. See TIO_DEV_CONNECT and TIO_DEV_CERTIFICATES command descriptions for the usage of this object.

Table 10. DATA_OBJECT_HEADER for SPDM Certificate Objects

Byte Offset	Bits	Name	Description
0h	31:0	DOBJ_ID	4h.
4h	31:0	DOBJ_LENGTH	Variable.
8h	15:0	DOBJ_VERSION	0: Minor. 1: Major.
Ah–Fh		–	Reserved.
10h	15:0	DEVICE_ID	PCIe Routing Identifier of function 0 of the device.
12h	7:0	SEGMENT_ID	PCIe Segment ID.
13h	7:0	CERT_TYPE	1h: SPDM certificate. 0h, 2h–FFh: Reserved.
14h–1Fh		–	Reserved.
20h	–	DOBJ_PAYLOAD	See Section 6.1.

4.5.5 SPDM Measurement Object

A data object containing the SPDM attestation report retrieved from a device. See TIO_DEV_MEASUREMENTS command description for the usage of this object.

Table 11. DATA_OBJECT_HEADER for SPDM Measurement Objects

Byte Offset	Bits	Name	Description
0h	31:0	DOBJ_ID	5h.
4h	31:0	DOBJ_LENGTH	Variable.
8h	15:0	DOBJ_VERSION	0: Minor. 1: Major.
Ah–Fh		–	Reserved.
10h	15:0	DEVICE_ID	PCIe Routing Identifier of function 0 of the device.
12h	7:0	SEGMENT_ID	PCIe Segment ID.
13h	7:0	MEAS_TYPE	1h: SPDM measurement. 0h, 2h–FFh: Reserved.
14h–1Fh		–	Reserved.
20h	–	DOBJ_PAYLOAD	See Section 6.2.

4.5.6 SPDM Interface Report Object

A data object containing the certificate chain retrieved from a device. See TIO_TDI_BIND and TIO_TDI_REPORT command descriptions for the usage of this object.

Table 12. DATA_OBJECT_HEADER for SPDM Measurement Objects

Byte Offset	Bits	Name	Description
0h	31:0	DOBJ_ID	6h.
4h	31:0	DOBJ_LENGTH	Variable.
8h	15:0	DOBJ_VERSION	0: Minor. 1: Major.
Ah–Fh		–	Reserved.
10h	15:0	DEVICE_ID	PCIe Routing Identifier of function 0 of the device.
12h	7:0	SEGMENT_ID	PCIe Segment ID.
13h	7:0	MEAS_TYPE	1h: TDISP interface report. 0h, 2h–FFh: Reserved.
14h–1Fh		–	Reserved.
20h	–	DOBJ_PAYLOAD	See Section 6.3.

Chapter 5 SPDM Transport

An SEV-TIO command can send SPDM requests to a device, receive an SPDM response from a device, and write objects like certificate chains and attestation reports to an output buffer.

5.1 SPDM Control Structure

Many SEV-TIO command buffer contains the SPDM control structure specified in Table 13.

Table 13. Layout of the SPDM_CTRL Structure

Byte Offset	Bits	Name	Description
0h	63:0	REQ_SLA	An SLA for the request buffer that contains an SPDM Request Object.
8h	63:0	RSP_SLA	An SLA for the response buffer that contains an SPDM Response Object.
10h	63:0	SCRATCH_SLA	An SLA for the scratch buffer that contains an SPDM Scratch Object.
18h	63:0	OUT_SLA	An SLA for the output buffer that may contain various data objects.

The request buffer, scratch buffer, and output buffer are written by SEV firmware. Each page of the request buffer and output buffer must be in the Firmware page state. Each page of the scratch buffer must be in the Scratch Buffer page state. The response buffer is read by SEV firmware and may be in any page state.

5.2 Control Flow

SEV-TIO introduces a new control flow scheme to support SPDM communication between SEV firmware and the device. Some commands might result in one or more SPDM request and responses transmitted between the SEV firmware and the device. For instance, the TIO_DEV_CONNECT command needs to trade many messages with the device to establish an SPDM connection and set up IDE streams. If a command might require SPDM communication, the command takes an SPDM_CTRL structure defined in Section 5.1.

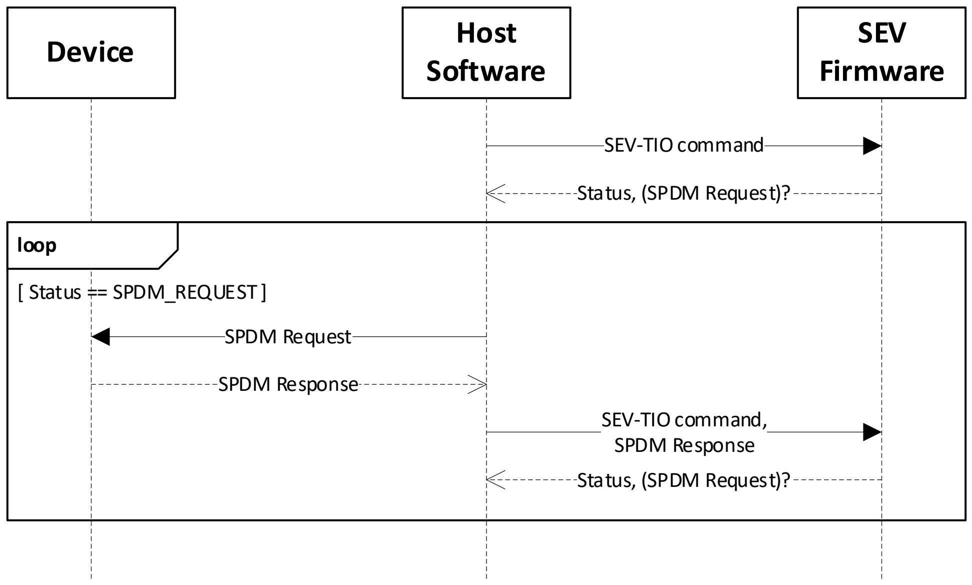

Figure 3: Control Flow for Commands that Send and Receive SPDM Messages

If a command invoked by host software needs to send an SPDM request, the command writes the request to the request buffer and returns the SPDM_REQUEST status code. When host software receives the SPDM_REQUEST status code, host software sends the message from the request buffer to the device. When the device returns a response, host software writes the response message into the response buffer and re-invokes the command so that SEV firmware can process the response. This process continues until the status code is no longer SPDM_REQUEST. Figure 3 illustrates the control flow. This sequence of reinvocation of commands is called an SPDM action.

Some SPDM actions produce output for host software to consume as part of the SPDM request. For instance, a command may output the certificate chain of the device. In this case, the output buffer is filled with the certificate chain.

When host software reinvokes a command after an SPDM_REQUEST status code, host software must preserve the contents of the scratch and output buffers.

An SPDM action operates on a single device. Only one SPDM action may be pending for a device at a time. However, SPDM actions for different devices may be interleaved. Commands that do not take SPDM_CTRL structures can be interleaved.

The SPDM transport layer may require knowledge of whether the transferred SPDM packets are secured. Host software can determine this via the SPDM_TYPE field of the SPDM Request Object defined in Section 4.5.1.

5.3 SPDM Buffer Capacity

The firmware may return SPDM_REQ_LENGTH, SPDM_SCRATCH_LENGTH, or SPDM_OUT_LENGTH to indicate that the response buffer, scratch buffer, or output buffer are too small, respectively. In this case, the firmware writes the desired size into the BUFFER_CAPACITY field of the buffer header. Host software should allocate at least that many bytes for the buffer. Host software must ensure that the original buffer contents are preserved on reallocation. Host software may leave the newly allocated portion of the buffer uninitialized.

Host software can invoke TIO_INFO to retrieve the REQ_LENGTH_HINT, SCRATCH_LENGTH_HINT, and OUT_LENGTH_HINT. These values indicate a heuristic for the sizes of each of these buffers. For instance, an implementation may return the maximum buffer size needed since the last SNP_INIT_EX invocation.

5.4 SPDM Scratch Buffer Page State

The first command invoked as part of an SPDM action transitions the provided data pages of the scratch buffer to the SPDM-Scratch page state. The SPDM-Scratch page state protects the contents of the SPDM scratch buffer and is intended to be accessed only by SEV firmware. When the firmware requests for host software to grow the buffer, the firmware transitions the newly added pages to the SPDM-Scratch page state.

When the last command returns with SUCCESS or an error other than SPDM_REQUEST, SPDM_REQ_LENGTH, SPDM_SCRATCH_LENGTH, or SPDM_OUT_LENGTH, the firmware transitions the pages of the SPDM scratch buffer back to the Firmware page state.

Chapter 6 Attestation Objects

TDISP capable devices provide attestation objects that allow the guest to validate the identity and configure of the device, and the configuration of the TDI it is bound to before allowing the TDI to access guest private memory. This section details the attestation object format that the SEV firmware outputs in the TIO_DEV_CERTIFICATES, TIO_DEV_MEASUREMENTS, and TIO_TDI_REPORT commands.

6.1 Certificates Object

TIO_DEV_CERTIFICATES returns a certificate chain retrieved from the device using the GET_CERTIFICATES SPDM request message. The SEV firmware returns the X.509 certificate chain provided by the device unaltered.

6.2 Measurements Object

TIO_DEV_MEASUREMENTS returns an attestation report retrieved from the device using the GET_MEASUREMENTS SPDM request message. All measurement blocks are requested by setting Param2 to FFh. Host software can request either the digest or the raw bitstream form of the attestation report.

The SEV firmware retrieves the attestation report through the cryptographically protected SPDM session. This provides the assurance that the attestation report came from the device and was not tampered with during transmission. Therefore, the SEV firmware does not request that the device sign the attestation report.

The measurement object is a MEASUREMENTS message as defined in the SPDM specification [6].

6.3 Interface Report Object

TIO_TDI_BIND and TIO_TDI_REPORT returns an interface report object. The report object is specified as defined in Table 15 of the PCI TDISP specification [3] with the exception that SEV firmware zeroes the MMIO base addresses (bytes 0 through 7 of each item in MMIO_RANGE) in the interface report. The SEV firmware zeroes these addresses because they are system physical addresses that are not useful to the guest. Additionally, system physical address space layout is typically sensitive to the host. Instead, the SEV firmware assists the guest in ensuring that the guest physical addresses of the MMIO are correctly mapped when the guest validates the range using the TIO_MSG_MMIO_VALIDATE_REQ guest message.

Chapter 7 Page States

This specification introduces new page states to manage special pages such as device context buffers, TDI context buffers, and scratch buffers.

7.1 Context-Guest

The SEV-SNP ABI defines the Context page state. This specification renames it to the Context-Guest page state to distinguish it from other context pages introduced below.

7.2 Context-Device

A Context-Device page state is used to store context information about the device. This page state has RMP.Immutable set to 1, RMP.Assigned set to 1, and RMP.ASID set to 0. The SEV firmware sets other fields in the RMP entry to distinguish the page from other firmware-owned pages. Context-Device pages are created with the TIO_DEV_CREATE command and are reclaimed with the TIO_DEV_RECLAIM command.

7.3 Context-TDI

A Context-TDI page state is used to store context information about the TDI. This page state has RMP.Immutable set to 1, RMP.Assigned set to 1, and RMP.ASID cleared to 0. The SEV firmware sets other fields in the RMP to distinguish it from other firmware-owned pages. Context-TDI pages are created with the TIO_TDI_CREATE command and are reclaimed with the TIO_TDI_RECLAIM command.

7.4 SPDM-Scratch

An SPDM-Scratch page state is to store SPDM scratch information during SPDM-related operations. This page state has RMP.Immutable set to 1, RMP.Assigned set to 1, and RMP.ASID cleared to 0. The SEV firmware sets other fields in the RMP to distinguish it from other firmware-owned pages. SPDM-Scratch pages are created on the first invocation of a command in an SPDM action and are reclaimed when the last command of an action is invoked.

Chapter 8 Command Reference

8.1 SNP_INIT_EX

The SNP_INIT_EX command is extended with a flag, TIO_EN, at bit 2 of offset 0h.

When INIT_RMP is 0, TIO_EN must be set to the same value provided in the SNP_INTI_EX invocation that originally initialized the RMP. If not, the firmware returns RMP_INIT_REQUIRED.

When INIT_RMP is 1 and TIO_EN is 1, the firmware initializes the hardware to support TIO. All TIO commands require that TIO_EN is 1.

8.2 SNP_PLATFORM_STATUS

The SNP_PLATFORM_STATUS command is extended to report the TIO_EN value provided to the SNP_INIT_EX invocation that initialized the RMP. The TIO_EN value is reported in bit 4 of offset 08h. The TIO_EN is set only when the STATE is INIT.

8.3 TIO_STATUS

The TIO_STATUS command returns status information about the SEV-TIO feature.

This command does not generate SPDM traffic with the device.

8.3.1 Parameters

Table 14. Layout of the CMD_TIO_STATUS Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length of this command buffer in bytes.
4h	31:0	–	–	Reserved.
8h	63:0	In	STATUS_PADDR	SPA of the TIO_STATUS structure.

Table 15. Layout of the TIO_STATUS Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length of this structure in bytes.
4h	31:1	–	Reserved.
	0	TIO_EN	Indicates if SEV-TIO is enabled.
8h	31:0	SPDM_REQ_SIZE_HINT	Hint for the SPDM request buffer size in bytes.
Ch	31:0	SPDM_SCRATCH_SIZE_HINT	Hint for the SPDM scratch buffer size in bytes.
10h	31:0	SPDM_OUT_SIZE_HINT	Hint for the SPDM output buffer size in bytes.
14h	31:0	DEVCTX_SIZE	Size of a device context buffer in bytes.
18h	31:0	TDICTX_SIZE	Size of a TDI context buffer in bytes.
1Ch	31:0	–	Reserved.

8.3.2 Actions

The firmware checks that STATUS_PADDR is a valid address, is 8-byte aligned, and does not cross a page boundary. If not, the firmware returns INVALID_ADDRESS. The firmware then checks that the page pointed at by STATUS_PADDR is a Firmware or Default page. If not, the firmware returns INVALID_PAGE_STATE.

The firmware writes out the status information structure to STATUS_PADDR.

8.3.3 Status Codes

Table 16. Status Codes for TIO_STATUS

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.

8.4 TIO_DEV_CREATE

Create a device context buffer that contains device-specific information related to SEV-TIO. The TIO_STATUS command returns the minimum size of the device context buffer.

The buffer may be reclaimed via the TIO_DEV_RECLAIM command.

This command does not generate SPDM traffic with the device.

8.4.1 Parameters

Table 17. Layout of the CMD_TIO_DEV_CREATE Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:0	–	–	Reserved.
8h	63:0	IN	DEV_CTX_SLA	A scatter list address pointing to a buffer to be used as a device context buffer.

8.4.2 Actions

The firmware checks that DEVCTX_SLA is a valid scatter list address. If not, the firmware returns INVALID_ADDRESS.

The firmware checks that the data pages of the buffer pointed at by DEV_CTX_SLA are all in the Firmware page state. If not, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the capacity of the provided buffer is large enough to hold a device context buffer. If not, the firmware returns BUFFER_LENGTH.

The firmware transitions each data page to the Context-Device page state and initializes the contents of the device context buffer.

8.4.3 Status Codes

Table 18. Status Codes for TIO_DEV_CREATE

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.

Status	Condition
INCORRECT_BUFFER_LENGTH	A provided buffer was too small.

8.5 TIO_DEV_RECLAIM

Reclaim a device context buffer that was created by TIO_DEV_CREATE.

This command does not generate SPDM traffic with the device.

8.5.1 Parameters

Table 19. Layout of the CMD_TIO_DEV_RECLAIM Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:0	–	–	Reserved.
8h	63:0	IN	DEV_CTX_SLA	Scatter list address of a device context buffer.

8.5.2 Actions

The firmware checks that DEV_CTX_SLA is a valid scatter list address. If not, the firmware returns INVALID_ADDRESS.

The firmware checks that the buffer is in the Device-Context page state. If not, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the buffer is a valid device context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware checks that all TDI context pages are reclaimed. If not, the firmware returns RECLAIM_REQUIRED.

The firmware reclaims any system resources associated with this device context. Then, the firmware transitions the data pages of the device context buffer to the Reclaim page state.

8.5.3 Status Codes

Table 20. Status Codes for TIO_DEV_RECLAIM

Status	Condition
SUCCESS	Successful completion.

Status	Condition
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_CONTEXT	Context page was invalid.
RECLAIM_REQUIRED	Resources must be reclaimed before invoking this command.

8.6 TIO_DEV_CONNECT

Request that the SEV firmware connect to the device by establishing a secure SPDM connection, set up IDE streams, and prepare the device for use as a TDISP device.

When the SPDM GET_CAPABILITIES message is sent during the construction of the SPDM session, the firmware sets CTExponent to 23, the maximum exponent allowed by PCI CMA. The firmware sets Flags, DataTransferSize, and MaxSPDMmsgSize in an implementation-specific manner.

When the NEGOTIATE_ALGORITHMS message is sent, the firmware supports the following cipher suites:

- **Signature**: RSA-SSA 3072, ECDSA P-256, ECDSA P-384
- **Hash**: SHA-256, SHA-384
- **Diffie Hellman**: ECDH P-256, ECDH P-384
- **AEAD**: AES-128-GCM, AES-256-GCM

The firmware retrieves the certificate chain from the slot provided by host software in CERT_SLOT and uses that certificate chain to authenticate the SPDM endpoint on the device. The firmware does not perform certificate chain validation but instead uses only the leaf key as if it is trustworthy. Each guest is responsible for retrieving the certificate chain and making its own decision to trust the device according to that certificate chain.

This command constructs one IDE selective stream between the device and the SoC per traffic class used by the device.

8.6.1 Parameters

Table 21. Layout of the CMD_TIO_DEV_CONNECT Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h–7h		–	–	Reserved.
8h–27h		IN	SPDM_CTRL	SPDM control structure defined in Section 5.1.

Byte Offset	Bits	In/Out	Name	Description
28h	15:0	IN	DEVICE_ID	The PCIe Routing Identifier of the device to connect to.
2Ah	15:0	IN	ROOT_PORT_ID	The PCIe Routing Identifier of the root port of the device.
2Ch	7:0	IN	SEGMENT_ID	The PCIe Segment Identifier of the device to connect to.
2Dh–2Fh		–	–	Reserved.
30h	63:0	IN	DEV_CTX_SLA	Scatter list address of the device context buffer.
38h	7:0	IN	TC_MASK	Bitmask of the traffic classes to initialize for SEV-TIO usage. Setting the *k*th bit of the TC_MASK to 1 indicates that the traffic class *k* will be initialized.
39h	7:0	IN	CERT_SLOT	Slot number of the certificate requested for constructing the SPDM session.
3Ah–3Fh		–	–	Reserved.

8.6.2 Actions

The firmware checks that DEV_CTX_SLA points to a valid device context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the SPDM_CTL structure is correctly formed. See Section 5.1 for the structure of the SPDM_CTL structure. If a page in one of the buffers is not in the required page state, the firmware returns INVALID_PAGE_STATE.

TC_MASK must have bit 0 set. If not, the firmware returns INVALID_PARAM.

The firmware checks that the device context buffer is in the Device-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the device context buffer is a valid device context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware establishes the SPDM connection, constructs an SPDM secure session, and negotiates and programs the IDE streams for the device. The firmware constructs one IDE stream for each Traffic Class indicated in TC_MASK. If the firmware determines that the device does not have sufficient capabilities to support an SPDM session, IDE streams, or TDISP functionality, the firmware returns INVALID_CONFIG.

After this command completes successfully, the SPDM output buffer contains the certificate chain of the device used to construct the SPDM secure session. The SHA-384 digest of this certificate chain is also stored within the device context page.

If this command fails, the device context buffer is returned to the state it was in before TIO_DEV_CONNECT was called. If and SPDM session was established before the error was detected, the firmware attempts to disable the IDE stream on the device and gracefully shut down the SPDM session so that the device is in its original state.

8.6.3 Status Codes

Table 22. Status Codes for TIO_DEV_CONNECT

Status	Condition
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_CONTEXT	Context page was invalid.
INVALID_PARAM	A parameter is invalid or incorrectly formed.
INVALID_CONFIG	The device is not sufficiently capable of supporting SPDM, IDE, or TDISP.

8.7 TIO_DEV_DISCONNECT

Disconnects the SEV firmware from the device by destroying the IDE streams associated with the device and shutting down the SPDM channel.

8.7.1 Parameters

Table 23. Layout of the CMD_TIO_DEV_DISCONNECT Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:0	–	–	Reserved.
8h–27h		IN	SPDM_CTRL	SPDM control structure defined in Section 5.1.
28h	63:0	IN	DEV_CTX_SLA	Scatter list address of the device context buffer.

8.7.2 Actions

The firmware checks that DEV_CTX_SLA points to a valid device context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the SPDM_CTL structure is correctly formed. See Section 5.1 for the structure of the SPDM_CTL structure. If a page in one of the buffers is not in the required page state, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the device context buffer is in the Device-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the device context buffer is a valid device context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware checks that this device has no bound TDIs. If the device has a bound TDI, the firmware returns IN_USE.

The firmware disables the IDE streams associated with this device and clears the keys in the root complex. Then, the firmware sends the END_SESSION SPDM request to gracefully close the SPDM channel.

If the device is no longer capable of responding to SPDM messages, host software can provide a SPDM response buffer scatter list pointer with the SPA field set to all 1s. When host software either provides a valid SPDM response packet or sets the scatter list point to all 1s, the command returns SUCCESS.

After this, the device context buffer may be either reclaimed with TIO_DEV_RECLAIM or it may be used to connect to another device with TIO_DEV_CONNECT.

8.7.3 Status Codes

Table 24. Status Codes for TIO_DEV_DISCONNECT

Status	Condition
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_CONTEXT	Context page was invalid.
INVALID_PARAM	A parameter is invalid or incorrectly formed.
IN_USE	A TDI is still bound to the device.

8.8 TIO_DEV_STATUS

Return information about the device status.

This status command outputs the number of TDI contexts that exist as well as one of the TDI context page SPAs. This mechanism can be used by host software to identify and reclaim all TDI pages associated with a device be iteratively invoking this command and TIO_TDI_RECLAIM until all TDI context pages are reclaimed. After this, this device context page can be reclaimed.

This command provides the REQUEST_PENDING, REQUEST_COMMAND, and REQUEST_TDI_PADDR status fields. These fields can inform host software of whether an SPDM request message is pending, from which command the request originated, and the TDI related to the request, if applicable. Note that only one SPDM request can be pending at a time for a given device.

This command does not generate SPDM traffic with the device.

8.8.1 Parameters

Table 25. Layout of the CMD_TIO_DEV_STATUS Structure

Byte Offset	Bits	In/Out	Name	Description
	31:0	IN	LENGTH	Length in bytes of this command buffer.
		–	–	Reserved. Must be zero.
	63:0	IN	DEV_CTX_PADDR	SPA of a device context page.
	31:0	IN	STATUS_LENGTH	Length in bytes of the status buffer.
	63:0	IN	STATUS_PADDR	SPA of the DEV_STATUS structure. See Table 26.

Table 26. Layout of the DEV_STATUS Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length in bytes of this structure in bytes.
4h	7:0	CTX_STATE	0: Not connected. 1: Connected.
5h	7:0	TC_MASK	Bitmask of initialized traffic classes. The *k*th bit of the TC_MASK to 1 indicates that the traffic class *k* is initialized for SEV-TIO.

Byte Offset	Bits	Name	Description
6h	7:2	–	Reserved.
	1	REQUEST_PENDING_TDI	If REQUEST_PENDING is 1, indicates that the pending request is associated with a TDI.
	0	REQUEST_PENDING	Flag indicating that the firmware has a pending SPDM message for this device.
7h	7:0	CERT_SLOT	SPDM SlotID for the certificate used to construct the SPDM channel.
8h	15:0	DEVICE_ID	PCIe Routing ID of the device.
Ah	7:0	SEGMENT_ID	PCIe Segment ID of the device.
Bh	7:0	TC_MASK	The traffic class mask provided to TIO_DEV_CONNECT.
Ch	15:0	REQUEST_PENDING_COMMAND	If REQUEST_PENDING is 1, then this field contains the command ID that is expecting an SPDM response.
10h – 1Bh		REQUEST_PENDING_INTERFACE_ID	If REQUEST_PENDING_TDI is 1, then this field contains the TDISP interface ID of the TDI.
1Ch	7:2	–	Reserved.
	1	NO_FW_UPDATE	Indicates if firmware updates be rejected by the device. This is 1 if any TDI is bound to a guest and the NO_FW_UPDATE flag was 1 in LOCK_INTERFACE_REQUEST.
	0	MEAS_DIGEST_VALID	Indicates that MEAS_DIGEST contains the digest of a device attestation report retrieved since TIO_DEV_CONNECT was invoked.
1Dh–1Fh		–	Reserved.
20h–2Fh		IDE_STREAM_ID[]	Array of 8 IDE stream IDs. Traffic class k is associated with IDE_STREAM_ID[k]. IDE_STREAM_ID[k] is valid only if bit k of TC_MASK is 1. Each IDE stream ID is 16 bits.
30h–5Fh		CERTS_DIGEST	Digest of the certificate chain used to construct the SPDM session.
60h–8Fh		MEAS_DIGEST	Digest of the latest measurement blocks retrieved by the SEV firmware. Valid only if MEAS_DIGEST_VALID is 1.

8.8.2 Actions

The firmware checks that DEV_CTX_SLA points to a valid device context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that STATUS_LENGTH is large enough to contain an DEV_STATUS structure. If not, the firmware returns INVALID_PARAM. The firmware then checks that the page is in the Firmware or Default page state. If not, the firmware returns INVALID_PAGE_STATE.

The firmware fills the status buffer with status information of the device.

This command never sends SPDM messages (and thus may be invoked even when SPDM messages are pending) to query the current state of the device.

8.8.3 Status Codes

Table 27. Status Codes for TIO_DEV_STATUS

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_CONTEXT	Context page was invalid.
INVALID_PARAM	A parameter is invalid or incorrectly formed.

8.9 TIO_DEV_MEASUREMENTS

Retrieve the device attestation report using the GET_MEASUREMENTS SPDM message.

The device attestation report is defined as the concatenation of all measurement blocks that can be retrieved by GET_MEASUREMENTS. A measurement block is specified by Section 10.11.1 of [SPDM].

Host software is expected to transfer the device attestation report to its guests. The guest can determine that the attestation is accurate and fresh by sending the TIO_TDI_INFO_REQ message to retrieve MEAS_DIGEST and MEAS_FRESH.

MEAS_DIGEST is the digest of the latest device attestation report that the firmware retrieved for this device. The guest can compare the digest of the report to MEAS_DIGEST to ensure that the report has not been modified.

MEAS_FRESH indicates that the device attestation report has been retrieved since the TDI was locked and bound to the guest. The guest can use this in conjunction with the NO_FW_UPDATE field in the TDI interface report to conclude that the device attestation report does not change while the TDI remains bound to the guest.

8.9.1 Parameters

Table 28. Layout of the CMD_TIO_DEV_MEASUREMENTS Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:1	–	–	Reserved.
	0	IN	RAW_BITSTREAM	0: Requests the digest form of the attestation report. 1: Requests the raw bitstream form of the attestation report.
8h–27h		IN	SPDM_CTRL	SPDM control structure defined in Section 5.1.
28h	63:0	IN	DEV_CTX_SLA	Scatter list address of the device context buffer.

8.9.2 Actions

The firmware checks that DEV_CTX_SLA points to a valid device context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the SPDM_CTL structure is correctly formed. See Section 5.1 for the structure of the SPDM_CTL structure. If a page in one of the buffers is not in the required page state, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the device context buffer is in the Device-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the device context buffer is a valid device context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware sends the GET_MEASUREMENTS request to the device to request all measurement blocks. The firmware fills the SPDM output buffer with an SPDM Measurement Object defined in Chapter 4. The firmware also stores the SHA-384 digest of the retrieved measurements in the device context page.

The firmware delegates all validation of the measurements to the guest.

8.9.3 Status Codes

Table 29. Status Codes for TIO_DEV_MEASUREMENTS

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_CONTEXT	Context page was invalid.
INVALID_PARAM	A parameter is invalid or incorrectly formed.

8.10 TIO_DEV_CERTIFICATES

Retrieve the certificate chain of the device. This retrieves the certificate of the slot used to construct the SPDM session.

8.10.1 Parameters

Table 30. Layout of the CMD_TIO_DEV_CERTIFICATES Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:0	–	–	Reserved.
8h–27h		IN	SPDM_CTRL	SPDM control structure defined in Section 5.1.
28h	63:0	IN	DEV_CTX_SLA	Scatter list address of the device context buffer.

8.10.2 Actions

The firmware checks that DEV_CTX_SLA points to a valid device context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the SPDM_CTL structure is correctly formed. See Section 5.1 for the structure of the SPDM_CTL structure. If a page in one of the buffers is not in the required page state, the firmware returns INVALID_PAGE_STATE.

The firmware checks that the device context buffer is in the Device-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the device context buffer is a valid device context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware sends the GET_CERTIFICATES request to the device to request the certificate chain of the device using the SlotID used to establish the SPDM connection. The firmware fills the SPDM output buffer with an SPDM Certificates Object defined in Chapter 4. The firmware also stores the SHA-384 digest of the retrieved certificates in the device context page, replacing any previously saved digest.

8.10.3 Status Codes

Table 31. Status Codes for TIO_DEV_CERTIFICATES

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_CONTEXT	Context page was invalid.
INVALID_PARAM	A parameter is invalid or incorrectly formed.

8.11 TIO_TDI_CREATE

Create a TDI context buffer that contains TDI-specific information related to SEV-TIO. The TIO_STATUS command returns the minimum size of the TDI context buffer.

The buffer may be reclaimed via the TIO_TDI_RECLAIM command.

This command does not generate SPDM traffic with the device.

8.11.1 Parameters

Table 32. Layout of the CMD_TIO_TDI_CREATE Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:0	–	–	Reserved.
8h	63:0	IN	DEV_CTX_SLA	Scatter list address of a device context buffer.
10h	63:0	IN	TDI_CTX_SLA	Scatter list address of a buffer to be used as a TDI context page.
18h–24h		In	INTERFACE_ID	Interface ID of the TDI as defined by TDISP.

8.11.2 Actions

The firmware checks that DEV_CTX_SLA points to a valid device context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the device context buffer is in the Device-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the device context buffer is a valid device context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware checks that TDI_CTX_SLA is a valid scatter list address. If not, the firmware returns INVALID_ADDRESS. The firmware checks that the data pages of the buffer pointed at by TDI_CTX_SLA are all in the Firmware page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the capacity of the provided TDI context buffer is large enough to hold a TDI context buffer. If not, the firmware returns INVALID_BUFFER_LENGTH.

The firmware checks that a TDI context has not already been created for the provided INTERFACE_ID. If a context buffer already exists, the firmware returns IN_USE.

The firmware transitions each data page of the TDI context buffer to the Context-TDI page state and initializes the contents of the TDI context buffer.

8.11.3 Status Codes

Table 33. Status Codes for TIO_TDI_CREATE

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_CONTEXT	Context page was invalid.
INVALID_PARAM	A parameter is invalid or incorrectly formed.
INVALID_BUFFER_LENGTH	The buffer was not large enough.
IN_USE	The TDI is already in use.

8.12 TIO_TDI_RECLAIM

Reclaim a TDI context page.

This command does not generate SPDM traffic with the device.

8.12.1 Parameters

Table 34. Layout of the CMD_TIO_TDI_RECLAIM Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:0	–	–	Reserved.
8h	63:0	IN	DEV_CTX_SLA	Scatter list address of a device context buffer.
10h	63:0	IN	TDI_CTX_SLA	Scatter list address of a TDI context buffer.

8.12.2 Actions

The firmware checks that DEV_CTX_SLA points to a valid device context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the device context buffer is in the Device-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the device context buffer is a valid device context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware checks that TDI_CTX_SLA points to a valid TDI context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the TDI context buffer is in the TDI-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the TDI context buffer is a valid TDI context buffer. If not, the firmware returns INVALID_CONTEXT.

Firmware checks that the TDI is not bound to a guest. If the TDI is bound to a guest, the firmware returns IN_USE.

The firmware transitions the page to the Reclaim page state.

8.12.3 Status Codes

Table 35. Status Codes for TIO_TDI_RECLAIM

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_CONTEXT	Context page was invalid.
INVALID_PARAM	A parameter is invalid or incorrectly formed.
IN_USE	The TDI is already bound to a guest.

8.13 TIO_TDI_BIND

Bind a TDI to a guest.

Host software uses this command to bind a guest and a TDI together and transition the TDI to the RUN state. After this command is complete, the guest can assess the authenticity and configuration of the device to determine whether the guest wants to use the TDI.

The GUEST_DEVICE_ID is the GDeviceID chosen by host software and given to the guest. The guest uses the GUEST_DEVICE_ID to refer to the TDI in guest requests. See TIO_GUEST_REQUEST for more information.

This command outputs the interface report to the SPDM output buffer as a side effect of the command. Host software is expected to provide this to the guest. Host software may also invoke the TIO_TDI_REPORT command to retrieve the report.

8.13.1 Parameters

Table 36. Layout of the CMD_TIO_TDI_BID Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:0	–	–	Reserved.
8h–27h		IN	SPDM_CTRL	SPDM control structure defined in Section 5.1.
28h	63:0	IN	DEV_CTX_SLA	Scatter list address of the device context buffer.
30h	63:0	IN	TDI_CTX_SLA	Scatter list address of a TDI context buffer.
38h	63:0	IN	GCTX_PADDR	System physical address of a guest context page.
40h	15:0		GUEST_DEVICE_ID	The GDeviceID that the guest uses to refer to the TDI.
42h	15:5			Reserved. Must be 0.
	4		ALL_REQUEST_REDIRECT	Requires ATS translated requests to route through the root complex. Must be 1.
	3		BIND_P2P	Enables direct P2P. Must be 0.
	2		LOCK_MSIX	Lock the MSI-X table and PBA.
	1		–	Reserved. Must be 0.
	0		NO_FW_UPDATE	Indicates that no firmware updates are allowed while the interface is locked.

44h	15:0	–	–	Reserved.

8.13.2 Actions

The firmware checks that DEV_CTX_SLA points to a valid device context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the device context buffer is in the Device-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the device context buffer is a valid device context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware checks that TDI_CTX_SLA points to a valid TDI context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the TDI context buffer is in the TDI-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the TDI context buffer is a valid TDI context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware checks that the SPDM_CTL structure is correctly formed. See Section 5.1 for the structure of the SPDM_CTL structure. If a page in one of the buffers is not in the required page state, the firmware returns INVALID_PAGE_STATE.

Firmware checks that the TDI is not bound to a guest. If the TDI is bound to a guest, the firmware returns IN_USE.

The firmware checks that GUEST_INTERFACE_ID is not used by another TDI already bound to this guest. If the ID is in use by another TDI for this guest, then the firmware returns IN_USE.

Then, the firmware transitions the TDI to the CONFIG_LOCKED state, passing the provided flags to the LOCK_INTERFACE_REQUEST message.

The firmware retrieves the interface report of the TDI and zeroes the addresses of the MMIO ranges in the report. The firmware then calculates and saves the digest of the interface report in the TDI context buffer.

Then, the firmware transitions the TDI to the RUN state.

After this command completes successfully, the SPDM output buffer contains the interface report for this TDI modified as described above.

8.13.3 Status Codes

Table 37. Status Codes for CMD_TIO_TDI_BIND

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_CONTEXT	Context page was invalid.
IN_USE	The guest device ID is already in use.

8.14 TIO_TDI_UNBIND

Unbind a TDI from the guest to which it is bound.

8.14.1 Parameters

Table 38. Layout of the CMD_TIO_TDI_UNBIND Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:0	–	–	Reserved.
8h–27h		IN	SPDM_CTRL	SPDM control structure defined in Section 5.1.
28h	63:0	IN	DEV_CTX_SLA	Scatter list address of the device context buffer.
30h	63:0	IN	TDI_CTX_SLA	Scatter list address of a TDI context buffer.
38h	63:0	IN	GCTX_PADDR	System physical address of a guest context page.

8.14.2 Actions

The firmware checks that DEV_CTX_SLA points to a valid device context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the device context buffer is in the Device-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the device context buffer is a valid device context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware checks that TDI_CTX_SLA points to a valid TDI context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the TDI context buffer is in the TDI-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the TDI context buffer is a valid TDI context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware checks that the SPDM_CTL structure is correctly formed. See Section 5.1 for the structure of the SPDM_CTL structure. If a page in one of the buffers is not in the required page state, the firmware returns INVALID_PAGE_STATE.

Firmware checks that the TDI is bound to the guest. If the TDI is not bound to the guest, the firmware returns INVALID_PARAM.

Firmware checks that each MMIO range in the interface report is in the Pre-Guest page state.

Firmware then transitions the TDI to the CONFIG_UNLOCKED state, regardless of what state the TDI was in at the time this command was invoked.

8.14.3 Status Codes

Table 39. Status Codes for TIO_TDI_UNBIND

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_CONTEXT	Context page was invalid.
INVALID_PARAM	A parameter is invalid or incorrectly formed.

8.15 TIO_TDI_REPORT

Retrieve the TDI interface report.

8.15.1 Parameters

Table 40. Layout of the CMD_TIO_TDI_REPORT Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:0	–	–	Reserved.
8h–27h		IN	SPDM_CTRL	SPDM control structure defined in Section 5.1.
28h	63:0	IN	DEV_CTX_SLA	Scatter list address of the device context buffer.
30h	63:0	IN	TDI_CTX_SLA	Scatter list address of a TDI context buffer.
38h	63:0	IN	GCTX_PADDR	System physical address of a guest context page.

8.15.2 Actions

The firmware checks that DEV_CTX_SLA points to a valid device context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the device context buffer is in the Device-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the device context buffer is a valid device context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware checks that TDI_CTX_SLA points to a valid TDI context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the TDI context buffer is in the TDI-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the TDI context buffer is a valid TDI context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware checks that the SPDM_CTL structure is correctly formed. See Section 5.1 for the structure of the SPDM_CTL structure. If a page in one of the buffers is not in the required page state, the firmware returns INVALID_PAGE_STATE.

Firmware checks that the TDI is bound to the guest. If the TDI is not bound to the guest, the firmware returns INVALID_PARAM.

The firmware retrieves the interface report of the TDI and zeroes the MMIO base addresses (bytes 0 through 7 of each item in MMIO_RANGE) in the interface report. See Section 2.9 for explanation for how guests use the interface report. The firmware then calculates and saves the digest of the interface report in the TDI context buffer.

After this command completes successfully, the SPDM output buffer contains the interface report for this TDI modified as described above.

8.15.3 Status Codes

Table 41. Status Codes for TIO_TDI_REPORT

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_CONTEXT	Context page was invalid.
INVALID_PARAM	A parameter is invalid or incorrectly formed.

8.16 TIO_TDI_INFO

Retrieve information about the TDI status.

This command does not generate SPDM traffic with the device.

8.16.1 Parameters

Table 42. Layout of the CMD_TIO_TDI_INFO Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:0	–	–	Reserved.
8h	63:0	IN	DEV_CTX_SLA	Scatter list address of the device context buffer.
10h	63:0	IN	TDI_CTX_SLA	Scatter list address of a TDI context buffer.
20h	31:0	IN	STATUS_LENGTH	Length of status structure in bytes.
24h	31:0	–	–	Reserved.
28h	63:0	IN	STATUS_PADDR	System physical address of TDI_STATUS structure. See Table 43.

Table 43. Layout of the TDI_INFO Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length of this structure in bytes.
4h–Fh		INTERFACE_ID	TDISP interface identifier.
10h	31:2	–	Reserved.
	1	MEAS_DIGEST_FRESH	0: The measurement was retrieved before TIO_TDI_BIND. 1: The measurement was retrieved after TIO_TDI_BIND.
	0	MEAS_DIGEST_VALID	0: MEAS_DIGEST is not valid because host software has not yet invoked TIO_DEV_MEASUREMENTS. 1: MEAS_DIGEST is valid.

Byte Offset	Bits	Name	Description
14h	15:5	–	Reserved.
	4	ALL_REQUEST_REDIRECT	Indicates whether ATS translated requests to route through the root complex is required.
	3	BIND_P2P	Indicates whether direct P2P is enabled.
	2	LOCK_MSIX	Indicates whether MSI-X table and PBA are locked.
	1	CACHE_LINE_SIZE	Indicates the cache line size. 0: 64B. 1: 128B.
	0	NO_FW_UPDATE	Indicates that no firmware updates are allowed while the interface is locked.
16h	15:0	–	Reserved.
18h–1Fh		SPDM_ALGOS	Algorithms used to establish the SPDM session. See Table 57 for the format of this field.
20h	383:0	CERTS_DIGEST	Digest of the certificate chain of the TDI.
50h	383:0	MEAS_DIGEST	Digest of the measurements of the TDI.
80h	383:0	INTERFACE_REPORT_DIGEST	Digest of interface report.
B0h–BFh		GUEST_REPORT_ID	Report ID of the guest that this device is assigned to. Valid only if CTX_STATE is 1.

8.16.2 Actions

The firmware checks that DEV_CTX_SLA points to a valid device context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the device context buffer is in the Device-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the device context buffer is a valid device context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware checks that TDI_CTX_SLA points to a valid TDI context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the TDI context buffer is in the TDI-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the TDI context buffer is a valid TDI context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware checks that STATUS_LENGTH is large enough to contain a TDI_STATUS structure. The firmware then checks that the pages that STATUS_PADDR point to are writeable by host software.

The firmware fills the status buffer with status information of the TDI.

This command never sends SPDM messages (and thus may be invoked even when SPDM messages are pending) to query the current state of the TDI.

8.16.3 Status Codes

Table 44. Status Codes for TIO_TDI_INFO

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_CONTEXT	Context page was invalid.
INVALID_PARAM	A parameter is invalid or incorrectly formed.

8.17 TIO_TDI_STATUS

Retrieves the state and other TDISP related information about a TDI from the device.

8.17.1 Parameters

Table 45. Layout of the CMD_TIO_TDI_STATUS Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:0	–	–	Reserved.
8h–27h		IN	SPDM_CTRL	SPDM control structure defined in Section 5.1.
28h	63:0	IN	DEV_CTX_SLA	Scatter list address of the device context buffer.
30h	63:0	IN	TDI_CTX_SLA	Scatter list address of a TDI context buffer.
38h	31:0	IN	STATUS_LENGTH	Length of status structure in bytes.
3Ch	31:0	–	–	Reserved.
40h	63:0	IN	STATUS_PADDR	System physical address of TDI_STATUS structure. See Table 43.

Table 46. Layout of the TDI_STATUS Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length of this structure in bytes.
4h	7:0	TDISP_STATE	0: CONFIG_UNLOCKED. 1: CONFIG_LOCKED. 2: RUN. 3: ERROR. *All other encodings reserved.*
5h–7h		–	Reserved.

8.17.2 Actions

The firmware checks that DEV_CTX_SLA points to a valid device context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the device context buffer is in the Device-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the device context buffer is a valid device context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware checks that TDI_CTX_SLA points to a valid TDI context page. If not, firmware returns INVALID_CONTEXT.

The firmware checks that the TDI context buffer is in the TDI-Context page state. If not, the firmware returns INVALID_PAGE_STATE. The firmware checks that the TDI context buffer is a valid TDI context buffer. If not, the firmware returns INVALID_CONTEXT.

The firmware checks that the SPDM_CTL structure is correctly formed. See Section 5.1 for the structure of the SPDM_CTL structure. If a page in one of the buffers is not in the required page state, the firmware returns INVALID_PAGE_STATE.

The firmware checks that STATUS_LENGTH is large enough to contain a TDI_STATUS structure. The firmware then checks that the pages that STATUS_PADDR point to are writeable by host software.

The firmware fills the TDI_INFO buffer with status information of the TDI retrieved from the device.

8.17.3 Status Codes

Table 47. Status Codes for TIO_TDI_STATUS

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_CONTEXT	Context page was invalid.
INVALID_PARAM	A parameter is invalid or incorrectly formed.

8.18 TIO_ASID_FENCE_CLEAR

Clears the ASID fencing for a guest after an I/O abort caused by an RMP check failure or a host page table fault. See Section 2.11 for information about ASID fencing.

This command does not generate SPDM traffic with the device.

8.18.1 Parameters

Table 48. Layout of the CMD_TIO_ASID_FENCE_CLEAR Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:0	–	–	Reserved.
8h	63:0	IN	GCTX_PADDR	System physical address of a guest context page.
10h	15:0	IN	DEVICE_ID	PCIe Routing ID of the root port that fenced the guest.
12h	7:0	IN	SEGMENT_ID	PCIe Segment ID of the root port that fenced the guest.
13h–1Fh		–	–	Reserved.

8.18.2 Actions

The firmware checks that GCTX_PADDR is a valid guest context page. If not, the firmware returns INVALID_GUEST.

The firmware checks that no TDIs are bound to the guest. If TDIs are bound to the guest, the firmware returns IN_USE.

The firmware clears the ASID fence from the identified root port.

8.18.3 Status Codes

Table 49. Status Codes for TIO_ASID_FENCE_CLEAR

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	A provided address was invalid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_CONTEXT	Context page was invalid.
INVALID_PARAM	A parameter is invalid or incorrectly formed.
INVALID_GUEST	Guest context page is invalid.

8.19 TIO_ASID_FENCE_STATUS

Returns the ASID fencing status for a guest. See Section 2.11 for information about ASID fencing.

This command does not generate SPDM traffic with the device.

8.19.1 Parameters

Table 50. Layout of the CMD_TIO_ASID_FENCE_STATUS Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:0	IN	ASID	ASID of a guest to retrieve status.
8h	63:0	IN	STATUS	SPA of an aligned 8B location to write the status. 0: ASID is not fenced. 1: ASID is fenced.
10h	15:0	IN	DEVICE_ID	PCIe Routing ID of the root port that fenced the guest.
12h	7:0	IN	SEGMENT_ID	PCIe Segment ID of the root port that fenced the guest.
13h–1Fh		–	–	Reserved.

8.19.2 Actions

The firmware checks that the STATUS address is a valid system physical address. If not, the firmware returns INVALID_ADDRESS.

The firmware writes to the system physical address STATUS indicating if the provided ASID is fenced by the provided root port.

8.19.3 Status Codes

Table 51. Status Codes for TIO_ASID_FENCE_STATUS

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	An address was invalid.

8.20 TIO_GUEST_REQUEST

Deliver an SEV-TIO related guest request to the firmware and receive the response.

Host software is expected to determine, through means outside of this interface, which TDI context page is associated with the guest request. For instance, the guest may provide host software with the guest request as well as the guest device ID that host software uses to look up the location of the TDI context page.

If host software provides the incorrect TDI context page to this command, the command returns a SUCCESS status code to the hypervisor and sends an error guest response message to the guest. This is to address the condition that the device has failed or has been removed from the system and the TDI context page is no longer available to communicate with. This also ensures that the guest's request sequence numbers remain synchronized with the firmware. If host software cannot locate a TDI context page because the device was removed, host software should provide INVALID_PADDR for TDI_CTX_PADDR.

8.20.1 Parameters

Table 52. Layout of the CMD_TIO_GUEST_REQUEST Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	IN	LENGTH	Length in bytes of this command buffer.
4h	31:0	–	–	Reserved.
8h–27h		IN	SPDM_CTRL	SPDM control structure defined in Section 5.1.
28h	63:0	IN	DEV_CTX_SLA	Scatter list address of the device context buffer.
30h	63:0	IN	TDI_CTX_SLA	Scatter list address of a TDI context buffer.
38h	63:0	IN	GCTX_PADDR	System physical address of a guest context page.
40h	63:0	In	GUEST_REQUEST_PADDR	Bits 63:0 of the sPA of the request message.
48h	63:0	In	GUEST_RESPONSE_PADDR	Bits 63:0 of the sPA of the response message.

8.20.2 Actions

Firmware performs the same actions of delivering a guest request as SNP_GUEST_REQUEST specified in [2] except that a device context buffer, TDI context buffer, and an SPDM control block are provided. This command may also send and receive SPDM messages with the device. Host software must ensure that the data pointed at by REQUEST_PADDR and RESPONSE_PADDR are preserved on each re-invocation of this command.

The key used to encrypt the TIO guest messages must be VMPCK0.

The SEV-TIO commands IDs are specified in Table 53.

Table 53. Message Type Encodings

Value	Message Type	Message Version
19	TIO_MSG_TDI_INFO_REQ	1
20	TIO_MSG_TDI_INFO_RSP	1
21	TIO_MSG_MMIO_VALIDATE _REQ	1
22	TIO_MSG_ MMIO_VALIDATE _RSP	1
23	TIO_MSG_MMIO_CONFIG_REQ	1
24	TIO_MSG_MMIO_CONFIG_RSP	1
25	TIO_MSG_ SDTE_WRITE_REQ	1
26	TIO_MSG_SDTE_WRITE_RSP	1

The semantics of each guest message are specified in Chapter 9.

8.20.3 Status Codes

Table 54. Status Codes for TIO_GUEST_REQUEST

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_PARAM	An invalid parameter was provided.
INVALID_ADDRESS	An address is invalid for use by the firmware is misaligned.
INVALID_GUEST	The provided guest context is invalid.
INVALID_GUEST_STATE	The guest is not in the correct state.
INVALID_PAGE_SIZE	A page was not the correct size.
INVALID_PAGE_STATE	A page was in the incorrect state.
AEAD_OFLOW	The message sequence number was incorrect, or the guest’s

Status	Condition
	message count would overflow.
BAD_MEASUREMENT	The message failed to authenticate.

Chapter 9 SEV-TIO Guest Messages

9.1 TDI Info

The guest sends this request to learn about the TDI.

Host software is expected to provide the guest the certificate chain, device measurements, and interface report through a protocol outside the scope of this interface. This message retrieves the digests of what the firmware saw of these objects. The guest can use these digests to determine whether host software tampered with the objects.

The guest can use MEAS_DIGEST_FRESH to determine that host software retrieved the device measurements object after this TDI was bound to the guest.

Table 55. Layout of the TIO_MSG_TDI_INFO_REQ Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length of this structure in bytes.
4h	15:0	GUEST_DEVICE_ID	Hypervisor provided identifier used by the guest to identify the TDI in guest messages.
5h–Fh		–	Reserved.

The firmware returns the TIO_TDI_INFO_RSP message to the guest filled with information about the TDI.

Table 56. Layout of the TIO_MSG_TDI_INFO_RSP Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length of this structure in bytes.
4h	15:0	STATUS	0: Success. 1: The TDI is not bound or unknown.
6h	15:0	GUEST_DEVICE_ID	Hypervisor provided identifier used by the guest to identify the TDI in guest messages.
8h	63:0	–	Reserved.

Byte Offset	Bits	Name	Description
10h	31:2	–	Reserved.
	1	MEAS_DIGEST_FRESH	0: The measurement was retrieved before TIO_TDI_BIND. 1: The measurement was retrieved after TIO_TDI_BIND.
	0	MEAS_DIGEST_VALID	0: MEAS_DIGEST is not valid because the host software has not yet invoked TIO_DEV_MEASUREMENTS. 1: MEAS_DIGEST is valid.
14h	15:5	–	Reserved.
	4	ALL_REQUEST_REDIRECT	Indicates if ATS translated requests to route through the root complex is required.
	3	BIND_P2P	Indicates if direct P2P is enabled.
	2	LOCK_MSIX	Indicates if MSI-X table and PBA are locked.
	1	CACHE_LINE_SIZE	Indicates the cache line size. 0: 64B. 1: 128B.
	0	NO_FW_UPDATE	Indicates that no firmware updates are allowed while the interface is locked.
16h	15:0	–	Reserved.
18h–1Fh		SPDM_ALGOS	Algorithms used to establish the SPDM session. See Table 57 for the format of this field.
20h	383:0	CERTS_DIGEST	Digest of the certificate chain of the TDI.
50h	383:0	MEAS_DIGEST	Digest of the measurements of the TDI.
80h	383:0	INTERFACE_REPORT_DIGEST	Digest of interface report.

Table 57. Layout of the SPDM_ALGOS Structure

Byte Offset	Bits	Name	Description
0h	7:0	DHE	0: secp256r1. 1: secp384r1. *All other encodings reserved.*

Byte Offset	Bits	Name	Description
1h	7:0	AEAD	0: AES-128-GCM. 1: AES-256-GCM *All other encodings reserved.*
2h	7:0	ASYM	0: TPM_ALG_RSASSA_3072. 1: TPM_ALG_ECDSA_ECC_NIST_P256. 2: TPM_ALG_ECDSA_ECC_NIST_P384. *All other encodings reserved.*
3h	7:0	HASH	0: TPM_ALG_SHA_256. 1: TPM_ALG_SHA_384. *All other encodings reserved.*
4h	7:0	KEY_SCHED	0: SPDM key schedule. *All other encodings reserved.*
5h–7h		–	Reserved.

9.2 MMIO Validate

This message asks the SEV firmware to sets the RMP.Validated bit to the guest-provided value for a subrange of an MMIO range. The SEV firmware also associates the range in the hardware with the specified TDI so that MMIO accesses are placed in the correct IDE stream.

Table 58. Layout of the TIO_MSG_MMIO_VALIDATE_REQ Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length of this structure in bytes.
4h	15:0	GUEST_DEVICE_ID	Hypervisor provided identifier used by the guest to identify the TDI in guest messages.
6h–Fh		–	Reserved.
10h	63:0	BASE	Guest physical address of the subrange.
18h	31:0	LENGTH	Length of the subrange in bytes.
1Ch	31:0	RANGE_OFFSET	Offset of the subrange within the MMIO range.

Byte Offset	Bits	Name	Description
20h	15:2	–	Reserved.
	1	FORCE_VALIDATED	0: If subrange does not have RMP.Validated set uniformly, fail. 1: If subrange does not have RMP.Validated set uniformly, force to requested value.
	0	VALIDATED	Desired value to set RMP.Validated for the range.
22h	15:0	RANGE_ID	RangeID of MMIO range.
24h	31:0	–	Reserved.

The firmware performs the following checks:

- The TDI is bound to the guest.
- The MMIO range ID is present in the interface report.
- All pages of the provided subrange are assigned to the guest.
- All pages of the provided subrange have the Immutable bit set in the RMP.
- If FORCE is 0, the Validated bit in the RMP is set uniformly for all pages of the provided subrange.
- The offset of the subrange matches the offset of the MMIO range in the interface report
- All pages of the provided MMIO range are I/O pages.

If any of the above checks fail, the firmware returns the appropriate error status code.

The firmware sets VALIDATED in the response to the value of the Validated bits of the MMIO range. The firmware then returns the message without altering the RMP.

The firmware sets the Validated bits in the RMP as requested by the guest. The firmware sets VALIDATED to the new value and sets CHANGED to 1 if the firmware changed the Validated bit and 0 otherwise.

The firmware responds with the TIO_MSG_MMIO_VALIDATE_RSP message.

Table 59. Layout of the TIO_MSG_MMIO_VALIDATE_RSP Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length of this structure in bytes.

Byte Offset	Bits	Name	Description
4h	31:0	STATUS	0: Success. 1: The TDI is not bound or unknown. 2: At least one page is not assigned to the guest. 3: At least one page is not mapped to the expected GPA. 4. The Validated bit is not uniformly set for the MMIO range. 5: At least one page is not an I/O page. 6: The provided MMIO range ID is not reported in the interface report. *All other encodings reserved.*
8h	63:0	GUEST_INTERFACE_ID	Hypervisor provided identifier used by the guest to identify the TDI in guest messages.
10h	63:0	BASE	Guest physical address of the subrange.
18h	31:0	LENGTH	Length of the subrange in bytes.
1Ch	31:0	RANGE_OFFSET	Offset of the subrange within the MMIO range.
20h	15:1	–	Reserved.
	0	CHANGED	Indicates that the Validated bit has changed due to this operation.
22h	15:0	RANGE_ID	Range of the MMIO.
24h	31:0	–	Reserved.

9.3 MMIO Configure

This message allows the guest to read and write TDISP-defined MMIO range attributes.

Table 60. Layout of the TIO_MSG_MMIO_CONFIG_REQ Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length of this structure in bytes.
4h	31:0	–	Reserved.
8h	15:0	GUEST_DEVICE_ID	Hypervisor provided identifier used by the guest to identify the TDI in guest messages.
Ah–Fh		–	Reserved.

Byte Offset	Bits	Name	Description
10h	31:16	RANGE_ID	Identifier of the range.
	15:3	–	Reserved.
	2	IS _NON_TEE_MEM	0: Can be mapped only into guest private memory. 1: Can be mapped into either guest private memory or shared memory. Ignored if WRITE is 0.
	1:0	–	Reserved.
4h	31:1	–	Reserved.
	0	WRITE	0: Only retrieve configuration of range. 1: Write configuration of range.

When WRITE is 0, the firmware returns a message with the current attributes of the MMIO range indicated by RANGE_ID. When WRITE is 1, the firmware sends the SET_MMIO_ATTRIBUTE_REQUEST TDISP message to the device to alter the configuration.

Table 61. Layout of the TIO_MSG_MMIO_CONFIG_RSP Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length of this structure in bytes.
4h	31:0	STATUS	0: Success. 1: The TDI is not bound or unknown. 2: The provided MMIO range ID is not reported in the interface report. 3: One or more attributes could not be changed. *All other encodings reserved.*
8h	15:0	GUEST_DEVICE_ID	Hypervisor provided identifier used by the guest to identify the TDI in guest messages.
Ah–Fh		–	Reserved.

Byte Offset	Bits	Name	Description
10h	31:16	RANGE_ID	Identifier of the range.
	15:4	–	Reserved.
	3	IS_MEM_ATTR_UPDATEABLE	Indicates certain TDISP flags can be updated. See TIO_TDI_MMIO_SET for details.
	2	IS _NON_TEE_MEM	0: Can be mapped only into guest private memory. 1: Can be mapped into either guest private memory or shared memory. Ignored if WRITE is 0.
	1	MSIX_PBA	Indicates if this range maps MSI-X PBA.
	0	MSX_TABLE	Indicates if the range maps MSI-X table.
4h	31:1	–	Reserved.
	0	WRITE	0: Only retrieve configuration of range. 1: Write configuration of range.

9.4 SDTE Write

Update the secure Device Table Entry (sDTE) for this TDI.

Table 62. Layout of the TIO_MSG_SDTE_WRITE_REQ Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length of this structure in bytes.
4h	15:0	GUEST_DEVICE_ID	Reserved.
8h	63:0	GUEST_INTERFACE_ID	Hypervisor provided identifier used by the guest to identify the TDI in guest messages.
10h	511:0	SDTE	sDTE to use to configure the guest-controlled fields. See Table 63.

Table 63. Layout of the SDTE Structure

Bits	Name	Description
511:350	–	Reserved.

Bits	Name	Description
351:321	VIRTUAL_TOM	vTOM applied to all TDI accesses.
320	VTOM_EN	0: vTOM not enabled. 1: vTOM enabled.
319:243	–	Reserved.
242:241	VMPL	VMPL applied to all accesses by this TDI.
240:1	–	Reserved.
0	V	0: TDI is not allowed to access guest private memory and this SDTE is not valid. 1: TDI is allowed to access guest private memory and this SDTE is valid.

The firmware sets the SDTE of the TDI to the value provided by the guest. The guest must send the TIO_MSG_TDI_INFO_REQ message before sending this message. This prevents the hypervisor from rebinding the TDI with the guest after the guest validated the attestation objects of the TDI.

Table 64. Layout of the TIO_MSG_SDTE_WRITE_RSP Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length of this structure in bytes.
4h	31:0	STATUS	0: Success. 1: The TDI is not bound or unknown. 2: Reserved fields were not 0. 3: A provided field has an invalid value. 4: The guest has not sent TIO_MSG_TDI_INFO_REQ since the last TIO_TDI_BIND command was invoked. *All other encodings reserved.*

Chapter 10 References

[1] Advanced Microdevices, "Secure Encrypted Virtualziation API," April 2020.

[2] Advanced Microdevices, "SEV Secure Nested Paging Firmware ABI Specification," 2022.

[3] PCI SIG, "TEE Device Interface Security Protocol (TDISP)," 2022.

[4] Advanced Microdevices, "AMD SEV-TIO: Trusted I/O for Secure Encrypted Virtualization," 2023.

[5] Advanced Microdevices, "AMD I/O Virtualization Technology (IOMMU) Specification," 2022.

[6] Distributed Management Task Force, "Security Protocol and Data Model (SPDM) Specification," 2022.