SEV-TIO Firmware Interface Specification

Publication #	**58271** Revision: **0.92**
Issue Date:	**August 2026**

Advanced Micro Devices 

Specification Agreement

This Specification Agreement (this "Agreement") is a legal agreement between Advanced Micro Devices, Inc. ("AMD") and "You" as the recipient of the attached AMD Specification (the "Specification"). If you are accessing the Specification as part of your performance of work for another party, you acknowledge that you have authority to bind such party to the terms and conditions of this Agreement. If you accessed the Specification by any means or otherwise use or provide Feedback (defined below) on the Specification, You agree to the terms and conditions set forth in this Agreement. If You do not agree to the terms and conditions set forth in this Agreement, you are not licensed to use the Specification; do not use, access or provide Feedback about the Specification.

In consideration of Your use or access of the Specification (in whole or in part), the receipt and sufficiency of which are acknowledged, You agree as follows:

1. You may review the Specification only (a) as a reference to assist You in planning and designing Your product, service or technology ("Product") to interface with an AMD product in compliance with the requirements as set forth in the Specification and (b) to provide Feedback about the information disclosed in the Specification to AMD.

2. Except as expressly set forth in Paragraph 1, all rights in and to the Specification are retained by AMD. This Agreement does not give You any rights under any AMD patents, copyrights, trademarks or other intellectual property rights. You may not (i) duplicate any part of the Specification; (ii) remove this Agreement or any notices from the Specification, or (iii) give any part of the Specification, or assign or otherwise provide Your rights under this Agreement, to anyone else.

3. The Specification may contain preliminary information, errors, or inaccuracies, or may not include certain necessary information. Additionally, AMD reserves the right to discontinue or make changes to the Specification and its products at any time without notice. The Specification is provided entirely "AS IS." AMD MAKES NO WARRANTY OF ANY KIND AND DISCLAIMS ALL EXPRESS, IMPLIED AND STATUTORY WARRANTIES, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NONINFRINGEMENT, TITLE OR THOSE WARRANTIES ARISING AS A COURSE OF DEALING OR CUSTOM OF TRADE. AMD SHALL NOT BE LIABLE FOR DIRECT, INDIRECT, CONSEQUENTIAL, SPECIAL, INCIDENTAL, PUNITIVE OR EXEMPLARY DAMAGES OF ANY KIND (INCLUDING LOSS OF BUSINESS, LOSS OF INFORMATION OR DATA, LOST PROFITS, LOSS OF CAPITAL, LOSS OF GOODWILL) REGARDLESS OF THE FORM OF ACTION WHETHER IN CONTRACT, TORT (INCLUDING NEGLIGENCE) AND STRICT PRODUCT LIABILITY OR OTHERWISE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

4. Furthermore, AMD's products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD's product could create a situation where personal injury, death, or severe property or environmental damage may occur.

5. You have no obligation to give AMD any suggestions, comments or feedback ("Feedback") relating to the Specification. However, any Feedback You voluntarily provide may be used by AMD without restriction, fee or obligation of confidentiality. Accordingly, if You do give AMD Feedback on any version of the Specification, You agree AMD may freely use, reproduce, license, distribute, and otherwise commercialize Your Feedback in any product, as well as has the right to sublicense third parties to do the same. Further, You will not give AMD any Feedback that You may have reason to believe is (i) subject to any patent, copyright or other intellectual property claim or right of any third party; or (ii) subject to license terms which seek to require any product or intellectual property incorporating or derived from Feedback or any Product or other AMD intellectual property to be licensed to or otherwise provided to any third party.

6. You shall adhere to all applicable U.S., European, and other export laws, including but not limited to the U.S. Export Administration Regulations ("EAR"), (15 C.F.R. Sections 730 through 774), and E.U. Council Regulation (EC) No 428/2009 of 5 May 2009. Further, pursuant to Section 740.6 of the EAR, You hereby certifies that, except pursuant to a license granted by the United States Department of Commerce Bureau of Industry and Security or as otherwise permitted pursuant to a License Exception under the U.S. Export Administration Regulations ("EAR"), You will not (1) export, re-export or release to a national of a country in Country Groups D:1, E:1 or E:2 any restricted technology,

software, or source code You receive hereunder, or (2) export to Country Groups D:1, E:1 or E:2 the direct product of such technology or software, if such foreign produced direct product is subject to national security controls as identified on the Commerce Control List (currently found in Supplement 1 to Part 774 of EAR). For the most current Country Group listings, or for additional information about the EAR or Your obligations under those regulations, please refer to the U.S. Bureau of Industry and Security's website at http://www.bis.doc.gov/.

7. If You are a part of the U.S. Government, then the Specification is provided with "RESTRICTED RIGHTS" as set forth in subparagraphs (c) (1) and (2) of the Commercial Computer Software-Restricted Rights clause at FAR 52.227-14 or subparagraph (c) (1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.277-7013, as applicable.

8. This Agreement is governed by the laws of the State of California without regard to its choice of law principles. Any dispute involving it must be brought in a court having jurisdiction of such dispute in Santa Clara County, California, and You waive any defenses and rights allowing the dispute to be litigated elsewhere. If any part of this agreement is unenforceable, it will be considered modified to the extent necessary to make it enforceable, and the remainder shall continue in effect. The failure of AMD to enforce any rights granted hereunder or to take action against You in the event of any breach hereunder shall not be deemed a waiver by AMD as to subsequent enforcement of rights or subsequent actions in the event of future breaches. This Agreement is the entire agreement between You and AMD concerning the Specification; it may be changed only by a written document signed by both You and an authorized representative of AMD.

Contents

List of Figures

List of Tables

Revision History

Date	Revision	Description
August 2026	0.92	Updates and Additions: • Added Section 1.4: References • Section 2.7: Description updated • Added Section 2.13: Virtual IOMMU • Added Section 2.14: Segmented RMP • Section 3.2: Reference to SEV-SNP status codes added • Added VERSION to all TIO host commands at Byte 7. • Section 7.1, 7.2, 7.3, 7.4: VIOMMU added • Section 7.7: IOMMU and Segmented RMP added • Section 7.9, 7.10, 7.16: Actions updated • Added Section 7.15: TIO_DEV_KEY_UPDATE • Section: 7.18: DOMAIN_ID, QUEUE_ID, and MMIO_REPORTING_OFFSET added • Section 7.23, 7.24: Status Codes updated • Added Section 7.26: TIO_VIOMMU_GUEST_INIT • Added Section 7.27: TIO_VIOMMU_GUEST_SHUTDOWN • Added Section 7.28: TIO_VIOMMU_STATUS • Section 7.29: Added VIOMMU messages, actions updated • Section 8.1, 8.2, 8.5, 8.6: GUEST_DEVICE_ID renamed to TDI_ID • Added Section 8.3: VIOMMU Configure • Added Section 8.4: VIOMMU MAPPING Configure • Section 8.6: New fields added to SDTE Structure, TIO_MSG_SDTE_WRITE_RSP updated
July 2025	0.91	Global updates related to revisions of firmware support for ABI (commands, return codes, etc.). Revision of Theory-of-Operation.
May 2023	0.70	Initial release.

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 provides a mechanism for guests to bind to and use trusted devices within their guest private address space.

1.2 Scope

This document describes 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 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.

1.4 References

Table 1. External References

Reference	Document
APM	*AMD64 Architecture Programmer's Manual, Volumes 1–5*, publication #40332
IOMMU	*AMD I/O Virtualization Technology (IOMMU) Specification,* publication #48882
SEV	*Secure Encrypted Virtualization API Specification*, publication #55766
SNP-ABI	*SEV Secure Nested Paging Firmware ABI Specification*, publication #56860

Chapter 2 Theory of Operation

This interface specification extends the Secure Encrypted Virtualization (SEV) and SEV Secure Nested Paging (SEV-SNP) firmware interface specifications (Advanced Microdevices, April 2020) (Advanced Microdevices, 2022) with a new set of host commands and SEV-TIO guest request messages, that allow a guest to establish trust in a device that supports TEE Device Interface Security Protocol (TDISP) (PCI SIG, 2022) 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 for supporting guests' assessment and usage of TDISP capable devices. For an overview of the feature, please see the AMD SEV-TIO whitepaper (Advanced Microdevices, 2023).

The AMD Security Processor (ASP) executes the SEV firmware that services all host commands and SEV-TIO 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 (Advanced Microdevices, 2022). To detect that the currently loaded SEV firmware supports SEV-TIO, software can retrieve bit 1 from Fn80000024_EBX_x00 using the FEATURE_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 MMIO pages in the platform to HV-fixed, which prevents host software from assigning MMIO pages to SNP 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 can 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 (IOMMU or x86), 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.

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 from the CONFIG_UNLOCKED state to the CONFIG_LOCKED state and to the RUN state. Upon transition to CONFIG_LOCKED state, firmware assigns a unique TDI_ID to the device (which can be retrieved from TDI_INFO data structure returned by the TIO_TDI_INFO command). 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_DIGEST_FRESH flag in the TIO_MSG_TDI_INFO_RSP guest message is 1. MEAS_DIGEST_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 causes 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 can 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 Validated bit. The guest can also clear the Validated bit for the MMIO range using this guest message.

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.

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_STATUS command can be used to retrieve the current state of the TDI to 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, then 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 all 1s.

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

2.12 Device Lifecycle

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

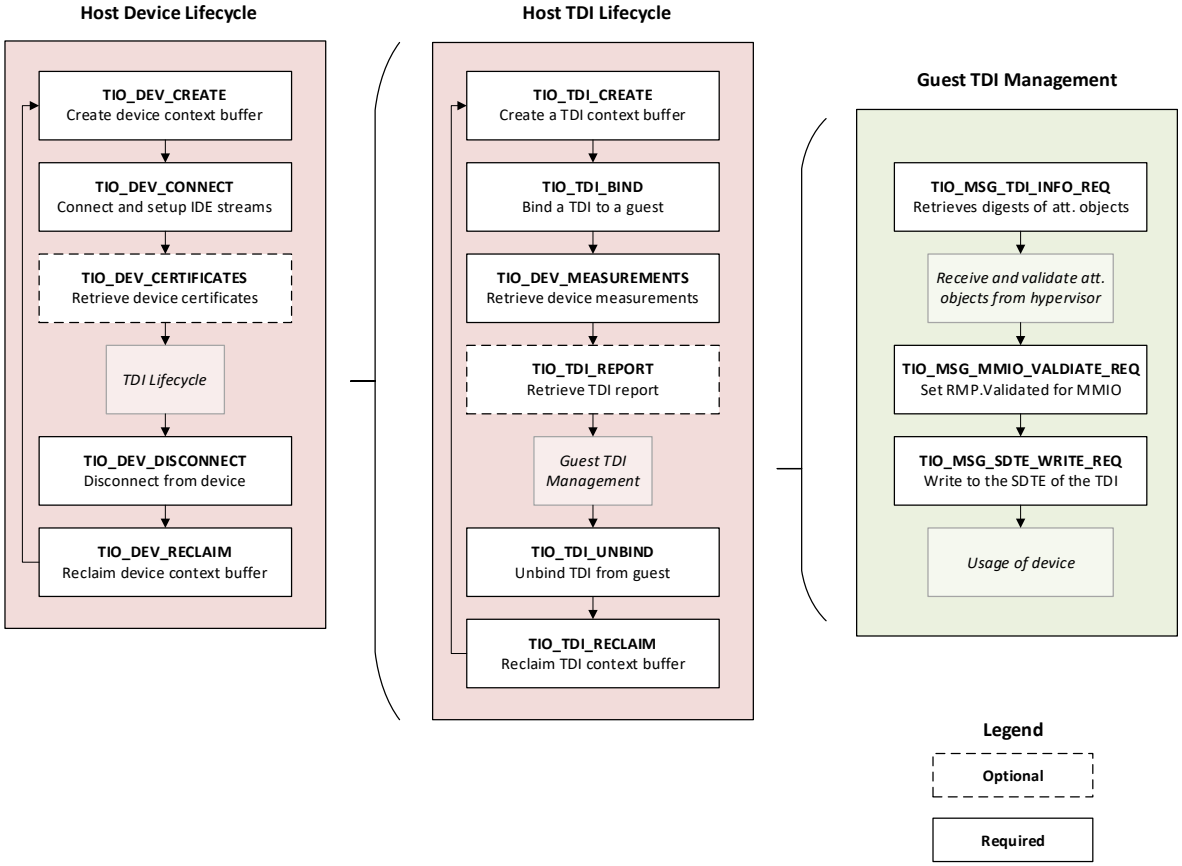

Figure 1. Lifecycle of a TDISP Capable Device

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.

2.13 Virtual IOMMU

The hypervisor can enable Virtual IOMMU (vIOMMU) functionality for any guest. vIOMMU is a hardware-enforced memory management functionality that allows the guest to directly configure and interact with the IOMMU, as opposed to the hypervisor intermediating interactions such as

commands, peripheral page request (PPR) logs and event logs. vIOMMU must be enabled in BIOS, and the hypervisor initializes the feature in the IOMMU as part of the SNP_INIT_EX options.

When vIOMMU is enabled, software must manage a new IOMMU private address space that contains IOMMU private pages. IOMMU private address space is defined in [IOMMU], Section 2.10.1.

2.14 Segmented RMP

Segmented RMP is a feature where the RMP is not treated as one monolithic structure but is instead broken into segments that can be allocated in non-contiguous locations in system physical memory. SEV-TIO requires that Segmented RMP is enabled. Software can enable the Segmented RMP feature in the Segmented RMP Configuration MSR (C001_0136h) defined in [APM], Section 15.36.22. The BIOS may also require configuration to enable Segmented RMP on some platforms.

Chapter 3 Mailbox Protocol

This specification extends the mailbox protocol with the SEV firmware defined in SEV ABI (Advanced Microdevices, April 2020) and SEV-SNP ABI (Advanced Microdevices, 2022).

3.1 Command Identifiers

The command identifiers for SEV-TIO related commands are specified in *Table 2 below*.

Table 2. Command Identifiers

Command	ID	Description
TIO_STATUS	D0h	Retrieve status of the SEV-TIO feature.
TIO_INIT	D1h	Initialize the TIO feature.
TIO_DEV_CREATE	D2h	Create a device context buffer.
TIO_DEV_RECLAIM	D3h	Reclaim a device context buffer.
TIO_DEV_CONNECT	D4h	Connect the SEV firmware to a device.
TIO_DEV_DISCONNECT	D5h	Disconnect the SEV firmware from a device.
TIO_DEV_STATUS	D6h	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_TDI_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 SEV-TIO 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.
TIO_TDI_DIGESTS_REPORT	E4h	The Host OS retrieves a TDI attestation report.

3.2 Status Codes

The status codes for SEV-TIO related commands are specified in *Table 3*. Additional codes can be found in Section 6.2 of [SNP-ABI]

Table 3. Status Codes

Status	Code	Description
UNSUPPORTED	15h	Feature is unsupported or invalid command version.
INCORRECT_BUFFER_LENGTH	30h	Buffer size does not match the buffer size provided by the TIO_STATUS command.
EXPAND_BUFFER_LENGTH_REQUEST	31h	Indicates that a firmware write buffer is not large enough. Indicates that the firmware is asking to increase the data buffer size. The new buffer size is specified in the BUFFER_CAPACITY field of BUFFER structure.
SPDM_REQUEST	32h	Indicates that firmware needs to send an SPDM message to the device, and the pending request is in the SPDM request buffer.
SPDM_ERROR	33h	Indicates an error in the SPDM channel.
INVALID_DATA_OBJECT	34h	Indicates that the data object is malformed.
ERROR_IN_DEV_CONNECTION	35h	Indicates that an unrecoverable error occurred in the SPDM secure session with the device.
INVALID_DEV_CTX	36h	Indicates that the provided page is not a valid device context page.
INVALID_TDI_CTX	37h	Indicates that the provided page is not a valid TDI context page.
INVALID_TDI	38h	Indicates that the identified TDI is invalid.
RECLAIM_REQUIRED	39h	Indicates that a reclaim of a context page is required before further use.
IN_USE	3Ah	The resource is in use.
INVALID_DEV_STATE	3Bh	The device state is invalid.
INVALID_TDI_STATE	3Ch	The TDI state is invalid.
DEV_CERT_CHANGED	3Dh	Digest of the certificate received in GET_CERTIFICATE does not match the previously saved digest.
RESYNC_REQUIRED	3Eh	The SPDM session needs to be resynced. This occurs when the SPDM generates resync required due to some inconsistency observed as defined in the spec.
RESPONSE_TOO_LARGE	3Fh	The response buffer is not large enough.

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 List Address (SLA) is a 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 4 below* describes the layout of an SLA.

Table 4. 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 5 below*.

Table 5. 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 an invalid entry, or it encounters the end of the page.

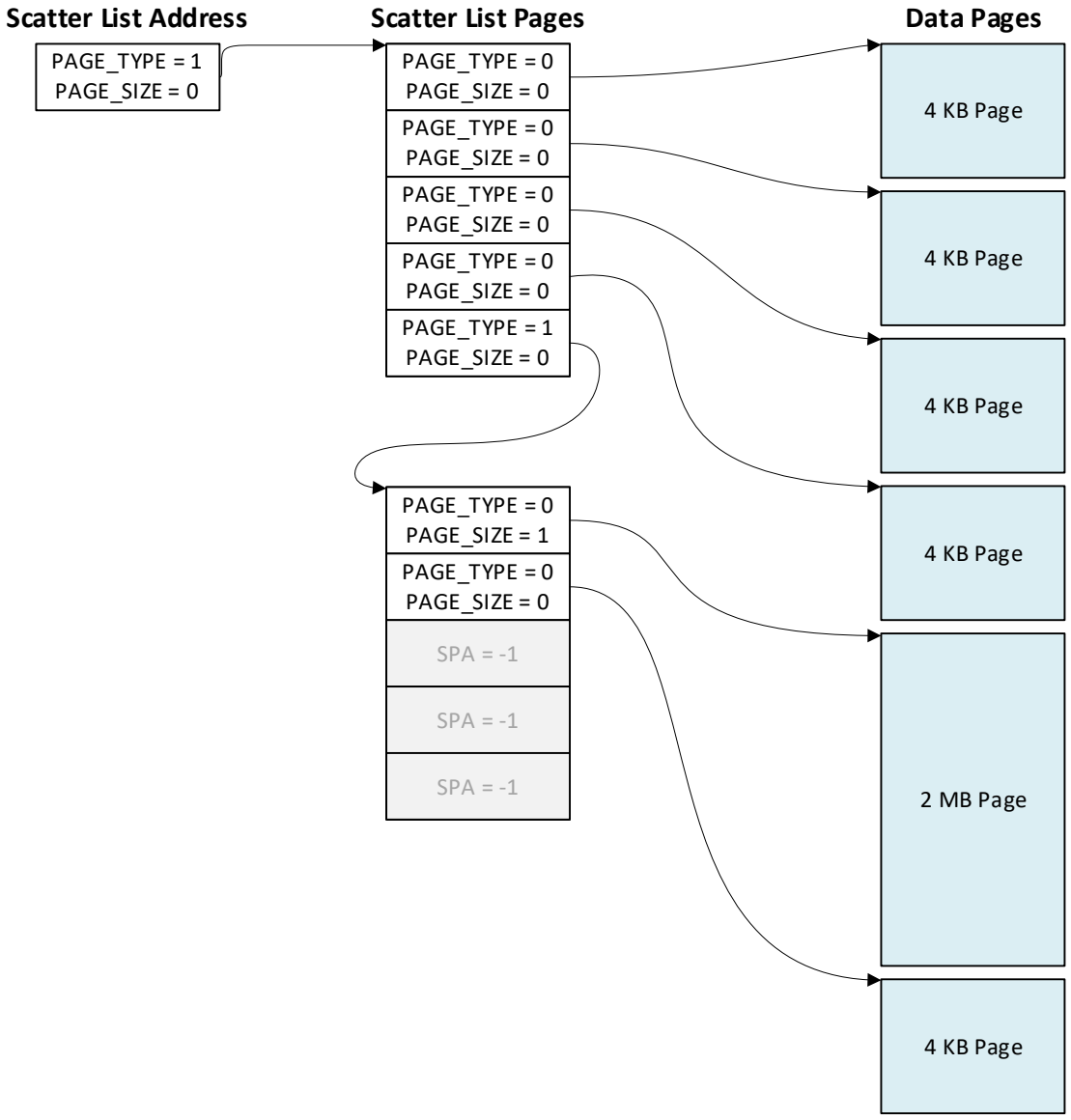

Figure 2. Scatter List and Its Formation of a Buffer

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 that 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 6 below* Table 6. Layout of the BUFFER Structurespecifies the structure of a logical buffer.

Table 6. 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_PAYLOAD_SIZE	The size of BUFFER_PAYLOAD in bytes.
8h	31:0	Reserved. Mbz.	
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_PAYLOAD_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.

4.5 Data Objects

Each data object stored in a logical buffer has a common header. *Table 7* below specifies the structure of the data object common header.

Table 7. 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 in bytes, including this header.
8h	15:0	DOBJ_VERSION	Version of the data object structure. Bit[7:0]: Minor version.

Byte Offset	Bits	Name	Description
			Bit[15:8]: Major version.

The following subsections specify each type of data object.

4.5.1 SPDM Request Object

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

Table 8. DATA_OBJECT_HEADER for SPDM Request Objects

Byte Offset	Bits	Name	Description
0h	31:0	DOBJ_ID	1h.
4h	31:0	DOBJ_LENGTH	Length of the data object in bytes, including this header.
8h	15:0	DOBJ_VERSION	Version of the data object structure. Bit[7:0]: Minor version. Bit[15:8]: Major version.
Ah	7:0	SPDM_TYPE	0x00: Invalid 0x01: SPDM message 0x02: Secured SPDM message 0x03-0xFF: reserved
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 a usage description of this object.

Table 9. DATA_OBJECT_HEADER for SPDM Response Objects

Byte Offset	Bits	Name	Description
0h	31:0	DOBJ_ID	2h. Unique identifier for SPDM response objects.
4h	31:0	DOBJ_LENGTH	Length of the data object in bytes, including this header.
8h	15:0	DOBJ_VERSION	Version of the data object structure. Bit[7:0]: Minor version. Bit[15:8]: Major version.
Ah	7:0	SPDM_TYPE	0x00: Invalid 0x01: SPDM message 0x02: Secured SPDM message 0x03-0xFF: reserved
Bh–Fh		–	Reserved.
10h	–	DOBJ_PAYLOAD	SPDM request packet.

4.5.3 SPDM Certificate Object

A data object containing the certificate chain that was 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	Length of the data object in bytes, including this header.
8h	15:0	DOBJ_VERSION	Version of the data object structure. Bit[7:0]: Minor version. Bit[15:8]: Major version.
Ah–Fh		–	Reserved.
10h	15:0	DEVICE_ID	PCIe Routing ID of function 0 of the device.
12h	7:0	SEGMENT_ID	PCIe Segment ID.
13h	7:0	CERT_TYPE	1h: SPDM certificate (see SPDM v1.2.1, Sec. 10.6.1, Table 28 for certificate format). 0h, 2h–FFh: Reserved.
14h–1Fh		–	Reserved.
20h	–	DOBJ_PAYLOAD	See *Certificates* Object *6.1*.

4.5.4 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	Length of the data object in bytes, including this header.
8h	15:0	DOBJ_VERSION	Version of the data object structure. Bit[7:0]: Minor version. Bit[15:8]: Major version.
Ah–Fh		–	Reserved.
10h	15:0	DEVICE_ID	PCIe Routing ID of function 0 of the device.
12h	7:0	SEGMENT_ID	PCIe Segment ID.
13h	7:0	MEAS_TYPE	1h: SPDM Measurement (see SPDM 1.2.1, Sec. 10.11, Table 43) 02h: SPDM Measurement Log L1/L2 (see SPDM 1.2.1, Sec. 10.11.2 for L1/L2 definition) 03h: SPDM Measurement Log L1/L2 and Signature

Byte Offset	Bits	Name	Description
			(see SPDM 1.2.1, Sec. 10.11.2 for signature generation) 0h, 04h-FFh: Reserved.
14h–1Fh		–	Reserved.
20h	–	DOBJ_PAYLOAD	See *Certificates* Object *6.2*.

4.5.5 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 Interface Objects

Byte Offset	Bits	Name	Description
0h	31:0	DOBJ_ID	6h.
4h	31:0	DOBJ_LENGTH	Length of the data object in bytes, including this header.
8h	15:0	DOBJ_VERSION	Version of the data object structure. Bit[7:0]: Minor version. Bit[15:8]: Major version.
Ah–Fh		–	Reserved.
10h	15:0	DEVICE_ID	PCIe Routing ID of function 0 of the device.
12h	7:0	SEGMENT_ID	PCIe Segment ID.
13h	7:0	MEAS_TYPE	1h: TDISP interface report. For more information, see Sec. 11.3.11 of the TDISP specification as included in PCIe Base Specification Rev 6.1. 0h, 2h–FFh: Reserved.
14h–1Fh		–	Reserved.
20h	–	DOBJ_PAYLOAD	See *Certificates* Object *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

Multiple SEV-TIO command buffers contain the SPDM control structure specified in

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 Firmware page state. The response buffer is read by SEV firmware and can 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 sets of requests and responses transmitted between the SEV firmware and the device. For instance, the TIO_DEV_CONNECT command needs to trade multiple messages with the device to establish an SPDM connection and set up IDE streams. If a command requires SPDM communications, then the command must include an SPDM_CTRL structure as defined in *Section 5.1*.

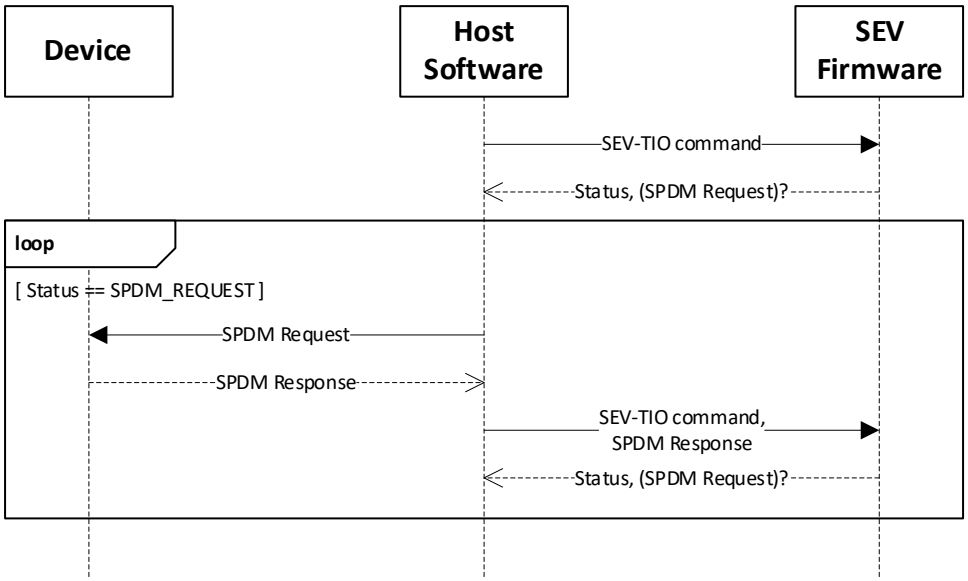

Figure 3. Control Flow for Commands that Send and Receive SPDM Messages

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_STATUS to retrieve the minimum and maximum buffer sizes for the request buffer, scratch buffer, output buffer, and response buffers. If software provides a buffer size outside of these bounds, the firmware will respond with status code INCORRECT_BUFFER_LENGTH.

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 the host software to grow the buffer, the firmware transitions the newly added pages to the SPDM-Scratch page state.

If the command returns SUCCESS or any unrecoverable error, 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 configuration of the device, as well as the configuration of the TDI the guest is bound to. This validation occurs prior to allowing the TDI to access guest private memory.

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. 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. Nevertheless, the SEV firmware always requests that the device sign the attestation report.

The measurement object is a MEASUREMENTS message as defined in the SPDM specification (Distributed Management Task Force, 2022).

6.3 Interface Report Object

TIO_TDI_BIND and TIO_TDI_REPORT return an interface report object. The report object is specified as defined in Table 15 of the PCI TDISP specification (PCI SIG, 2022). 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 Command Reference

This chapter provides information regarding the SEV-SNP modified commands, and the additional SEV-TIO commands. The status codes returned for these commands are specific to TIO; for all other SEV status codes (e.g., GUEST_INVALID) please refer to the SEV-SNP specification.

***NOTE**: Any command with LENGTH as an input field will be checked to match the specific size. INVALID_LENGTH will be returned if the input command length does not match the version's specific length.*

***NOTE**: All TIO commands as of v0.92 of this specification contain VERSION as an input field at byte offset 0x7. The next version of this specification will introduce a new ABI to query the range of supported versions for each command.*

7.1 SNP_INIT_EX

The SNP_INIT_EX command is extended with a flag, TIO_EN, at bit 4 of offset 0h.

When INIT_RMP is 0, TIO_EN must be set to the same value provided in the SNP_INIT_EX invocation that originally initialized the RMP. If not, 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 TIO_EN is 1. Software can invoke the TIO_INIT command after the SNP_INIT_EX command to enable SEV-TIO in the IOMMUs and PCIe root ports.

The SNP_INIT_EX command is also extended with the flag, VIOMMU_EN at bit 5 of offset 0h. In order to enable the VIOMMU feature firmware will verify that VIOMMU_EN is set to 1. As with TIO_EN, subsequent SNP_INIT_EX invocations when INIT_RMP is 0 must have the same VIOMMU_EN setting.

7.2 SNP_PLATFORM_STATUS

The SNP_PLATFORM_STATUS command is extended to report the IS_TIO_EN, IS_VIOMMU_EN, and IS_TIO_INIT values.

7.3 SNP_DECOMMISSION

The SNP_DECOMMISSION command is extended to check that no TDIs are bound to the guest. If a TDI is bound to a guest, the command fails with IN_USE. Software can invoke TIO_TDI_UNBIND on all TDIs bound to the guest to decommission the guest. Firmware will verify whether VIOMMU instances exist for the guest. If so, then firmware will return the IN_USE status code will fail the command.

7.4 SNP_SHUTDOWN

The SNP_SHUTDOWN command is extended to require that all devices are disconnected via the TIO_DEV_DISCONNECT command before SNP_SHUTDOWN or SNP_SHUTDOWN_EX succeeds. If a device is not yet disconnected, the SNP_SHUTDOWN and SNP_SHUTDOWN_EX commands return RECLAIM_REQUIRED.

If IOMMU_SHUTDOWN is set in SNP_SHUTDOWN, then all active IOMMUs will be shut down, vIOMMU state will be reset, and all vIOMMU resources (backing pages, device mapping tables, and domain mapping tables) will be transitioned to the Reclaim page state.

7.5 TIO_TDI_DIGEST_REPORT

The Host OS can directly request that the firmware construct a TDI attestation report.

The firmware generates a TDI attestation report by generating a Header, a “Guest Binding Tuple” (TUPLE) and a signature then placing the components into the specified Output buffer for the Host OS.

7.5.1 Guest (TVM) Identifier

There is a requirement to support a Guest (TVM) Identifier. This identifier is used to provide a logical binding of a guest (TVM) and device (TDI). The Guest Identifier is composed of the REPORT_ID field from the Guest ATTESTATION_REPORT structure and is included in the TDI Attestation Report.

7.5.2 Guest Binding Tuple (TUPLE)

The Guest Binding Tuple (TUPLE) is created by firmware and provides a cryptographic binding of the digests (hashes) of the following 3 objects:

- device certification
- measurements
- interface report information

Then the plaintext Guest (TVM) Identifier (REPORT_ID) is added.

Header

The firmware generates a header consisting of two 32-bit fields. The first field is set to 0, and the second is set to the current version (1h).

Guest Binding Tuple (TUPLE) Generation

The Guest Binding Tuple (TUPLE) provides a binding of four objects to bind the guest and device cryptographically together.

The firmware uses the SHA-384 digests (hashes) previously generated to form the TUPLE:

- CertDigest := sha384 digest(leaf certificate retrieved via GET_CERTIFICATES)
- DevAttDigest := sha384 digest(device attestation report retrieved via GET_MEASUREMENTS)
- DevIntReportDigest := sha384 digest(interface report retrieved via INTERFACE_REPORT_REQUEST)

The REPORT_ID value (Guest Identifier) is added to the TUPLE as the fourth member.

- REPORT_ID (Guest Identifier)

TUPLE: All non-SIGNATURE fields listed in *Table 15 below*.

SIGNATURE: = Signature(TUPLE, GuestAttKey)
GuestAttKey is the same key used to sign attestation reports (either VCEK or VLEK).

Signature

The TIO_TDI_DIGESTS_REPORT command has the firmware generate the Guest Binding TUPLE, adds a header object, and then signs both object sets.

The firmware signs the non-SIGNATURE fields in *Table 15 below* with either the VLEK or the VCEK. If VLEK is installed it signs with VLEK, otherwise it signs with VCEK. The Signature is placed into the SIGNATURE field of the SNP_TDI_REPORT_REQ Return Structure.

7.5.3 Parameters

Table 14. Layout of the CMD_TIO_TDI_DIGEST_REPORT Request Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length in bytes of this command buffer.
4h	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
8h	63:0	In	GCTX_PADDR	Bits 63:00 of the sPA of the guest context page.
10h	63:0	In	DEV_CTX_SLA	Scatter list address of the device context buffer.
18h	63:0	In	TDI_CTX_SLA	Scatter list address of the TDI context buffer.
20h–27h		In	TDI_DIGEST_PADDR	Location for firmware to write the report.
28h	31:0	In	FLAGS 31:2 Reserved 1:0 KEY_SEL	KEY_SEL: 0: If VCEK installed sign with VCEK otherwise sign with VLEK 1: Sign with VCEK 2: Sign with VLEK 3: Reserved
2Ch–3Fh		–	–	Reserved.

Table 15. TIO_TDI_DIGEST_REPORT Return Structure

Byte Offset	Bits	Name	Description
0h	0:31	ZERO_FIELD	Mbz. Must be set to 0.

Byte Offset	Bits	Name	Description
4h	0:7	VERSION	Set to 1h for this revision.
5h	0:23	-	Reserved.
8h	0:63	-	Reserved. Mbz.
10h	0:383	TUPLE_CERT_DIGEST	SHA-384 digest (hash) of the leaf certificate retrieved via GET_CERTIFICATES.
40h	0:383	TUPLE_MEAS_DIGEST	SHA-384 digest (hash) of the device attestation report retrieved via GET_MEASUREMENTS.
70h	0:383	TUPLE_INTER_DIGEST	SHA-384 digest (hash) of the interface report retrieved via INTERFACE_REPORT_REQUEST
A0h	0:63	TDI_REPORT_COUNT	TDIReportCount value in the TDI context.
A8h–AFh		–	Reserved.
B0h	0:255	TUPLE_GUEST_ID	REPORT_ID value from the guest attestation report
D0h	0:4095	SIGNATURE	VCEK or VLEK signature of the Header fields and the TUPLE values (from byte offset 0x0).

7.5.4 Actions

The firmware checks that the platform is in the INIT state. If not, firmware returns INVALID_PLATFORM_STATE.

The firmware checks that the command buffer is aligned to a 16-byte boundary.

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

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

The firmware checks that TDI_CTX_SLA points to a valid scatter list address. If not, firmware returns INVALID_ADDRESS.

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

The firmware checks that the TDI_DIGEST_PADDR is 8-byte aligned and does not cross a 4k page boundary. If not, firmware returns INVALID_ADDRESS. The firmware then checks that the page is in the Firmware or Default page state. If not, firmware returns INVALID_PAGE_STATE.

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

7.5.5 Status Codes

Table 16. Status Codes for TIO_TDI_DIGEST_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_PLATFORM_STATE	Not in the INIT state.
INVALID_TDI_STATE	The TDI state is invalid.
INVALID_DEV_STATE	The device state is invalid.

7.6 TIO_STATUS

The TIO_STATUS command returns status information about the SEV-TIO feature. SNP does not need to be initialized. This command does not generate SPDM traffic with the device.

7.6.1 Parameters

Table 17. 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	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
8h	63:0	In	STATUS_PADDR	SPA of the TIO_STATUS structure.

Table 18. Layout of the TIO_STATUS Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length of this structure in bytes.
4h	31:2	–	Reserved.
	1	TIO_INIT_DONE	Indicates TIO_INIT has been invoked.
	0	TIO_EN	Indicates if SEV-TIO is enabled.
8h	31:0	SPDM_REQ_SIZE_MIN	Minimum SPDM request buffer size in bytes.
Ch	31:0	SPDM_REQ_SIZE_MAX	Maximum SPDM request buffer size in bytes.
10h	31:0	SPDM_SCRATCH_SIZE_MIN	Minimum SPDM scratch buffer size in bytes.
14h	31:0	SPDM_SCRATCH_SIZE_MAX	Maximum SPDM scratch buffer size in bytes.
18h	31:0	SPDM_OUT_SIZE_MIN	Minimum SPDM output buffer size in bytes.
1Ch	31:0	SPDM_OUT_SIZE_MAX	Maximum SPDM output buffer size in bytes.
20h	31:0	SPDM_RSP_SIZE_MIN	Minimum SPDM response buffer size in bytes.
24h	31:0	SPDM_RSP_SIZE_MAX	Maximum SPDM response buffer size in bytes.
28h	31:0	DEVCTX_SIZE	Size of a device context buffer in bytes.
2Ch	31:0	TDICTX_SIZE	Size of a TDI context buffer in bytes.
30h-33h		TIO_CRYPTO_ALG	0x00: SHA2-384

Byte Offset	Bits	Name	Description
			0x01 – 0x33: Reserved
34h-3Fh		–	Reserved.

7.6.2 Actions

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

The firmware writes out the status information structure to STATUS_PADDR.

7.6.3 Status Codes

Table 19. 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.

7.7 TIO_INIT

Initialize the SEV-TIO feature. Successful completion of TIO_INIT requires an earlier invocation of the SNP_INIT_EX command with the TIO_EN flag set.

After successful completion of this command, the SEV-TIO feature will be enabled in the IOMMUs and PCIe root ports. It will remain enabled until the SNP_SHUTDOWN_EX command is invoked with the IOMMU_SHUTDOWN flag set which will disable both SEV-SNP and SEV-TIO in the IOMMUs and PCIe root complexes.

7.7.1 Parameters

Table 20. Layout of the TIO_INIT Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length of this command buffer in bytes.
4h	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
8h-Fh		–	–	Reserved.

7.7.2 Actions

The firmware checks that SNP_INIT_EX was invoked with the TIO_EN flag set. If not, firmware returns INVALID_PLATFORM_STATE.

The firmware checks that bit 0 (SegRmpEn) in the Segmented RMP Configuration MSR (C001_0136h) is set. If not, firmware returns INVALID_PLATFORM_STATE.

The firmware enables TIO in the I/O subsystem of the SoC.

If VIOMMU_EN was set to 1 in the SNP_INIT_EX options, then firmware will configure each IOMMU to initialize the VIOMMU feature. If the firmware fails to configure IOMMU due to hardware failure, firmware returns UNSUPPORTED.

7.7.3 Status Codes

Table 21. Status Codes for TIO_INIT

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	SNP_INIT_EX was not invoked with TIO_EN set.
UNSUPPORTED	IOMMU failed to configure.

7.8 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.

7.8.1 Parameters

Table 22. Layout of the CMD_TIO_DEV_CREATE Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length of this command buffer in bytes.
4h	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
8h	63:0	In	DEV_CTX_SLA	A scatter list address pointing to a buffer to be used as a device context buffer.
10h	15:0	In	DEVICE_ID	The PCIe Routing ID of the device to connect to.
12h	15:0	In	ROOT_PORT_ID	The PCI Express Port number for the given PCI Express Link of the root port.
14h	7:0	In	SEGMENT_ID	The PCIe domain Segment Identifier of the device to connect to. This SEGMENT_ID is a PCI domain segment ID.
15h-1Fh		–	–	Reserved.

7.8.2 Actions

The firmware checks if TIO is enabled and is initialized. If not, firmware returns INVALID_CONFIG.

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

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

7.8.3 Status Codes

Table 23. 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.
INCORRECT_BUFFER_LENGTH	A provided buffer was too small.
INVALID_CONFIG	The TIO device is either not enable and initialized.
IN_USE	The resource is in use.

7.9 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.

7.9.1 Parameters

Table 24. 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	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
8h	63:0	In	DEV_CTX_SLA	Scatter list address of a device context buffer.

7.9.2 Actions

The firmware checks that the platform is in the INIT state. If not, firmware returns INVALID_PLATFORM_STATE.

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

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

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

The firmware checks that the device is not connected. If not, firmware returns INVALID_DEV_STATE.

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.

7.9.3 Status Codes

Table 25. Status Codes for TIO_DEV_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.
RECLAIM_REQUIRED	Resources must be reclaimed before invoking this command.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_DEV_STATE	The device is connected.

7.10 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-PSSSA 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.

7.10.1 Parameters

Table 26. 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	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
8h-27h		In	SPDM_CTRL	SPDM control structure defined in *Section 5.1*.
28h-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	7:1	–	–	Reserved.
	0	In	ATC_FLAG	0: non-ATC device 1: ATC (Address Translation Cache) enabled device
3Bh-3Fh		–	–	Reserved.
40h	7:0	In	IDE_STREAM_ID0	IDE stream ID 0 to be associated with this device. Valid only if TC_MASK bit 0 is set.
41h	7:0	In	IDE_STREAM_ID1	IDE stream ID 1 to be associated with this device. Valid only if TC_MASK bit 1 is set.
42h	7:0	In	IDE_STREAM_ID2	IDE stream ID 2 to be associated with this device. Valid only if TC_MASK bit 2 is set.
43h	7:0	In	IDE_STREAM_ID3	IDE stream ID 3 to be associated with this device. Valid only if TC_MASK bit 3 is set.
44h	7:0	In	IDE_STREAM_ID4	IDE stream ID 4 to be associated with this device. Valid only if TC_MASK bit 4 is set.
45h	7:0	In	IDE_STREAM_ID5	IDE stream ID 5 to be associated with this device. Valid only if TC_MASK bit 5 is set
46h	7:0	In	IDE_STREAM_ID6	IDE stream ID 6 to be associated with this device. Valid only if TC_MASK bit 6 is set.
47h	7:0	In	IDE_STREAM_ID7	IDE stream ID 7 to be associated with this device. Valid only if TC_MASK bit 7 is set.
48h-4Fh		–	–	Reserved.

7.10.2 Actions

The firmware checks if TIO is enabled and is initialized. If not, firmware returns INVALID_PLATFORM_STATE.

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

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, firmware returns INVALID_PAGE_STATE.

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

The firmware checks that the device context buffer is in the Device-Context page state. If not, firmware returns INVALID_PAGE_STATE. The firmware checks that the device context buffer is a valid device context buffer. If not, firmware returns INVALID_CONTEXT. Firmware will record the value of ATC_FLAG in the device 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 selective 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, firmware returns INVALID_CONFIG.

After this command successfully completes, 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 an SPDM session was established before the error was detected, the firmware resets the internal firmware host device state to the previous state prior to the TIO_DEV_CONNECT command.

7.10.3 Status Codes

Table 27. 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.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.

Status	Condition
INCORRECT_BUFFER_LENGTH	Buffer size does not match the buffer size provided by the TIO_STATUS command.
EXPAND_BUFFER_LENGTH_REQUEST	Indicates that a firmware write buffer is not large enough. Indicates that the firmware is asking to increase the data buffer size. The new buffer size is specified in the BUFFER_CAPACITY field of BUFFER structure.
SPDM_REQUEST	Indicates that firmware needs to send an SPDM message to the device, and the pending request is in the SPDM request buffer.
SPDM_ERROR	Indicates an error in the SPDM channel.
RESYNC_REQUIRED	The SPDM session needs to be resynced. This occurs when the SPDM generates resync required due to some inconsistency observed as defined in the spec.
RESPONSE_TOO_LARGE	The response buffer is not large enough.

7.11 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.

7.11.1 Parameters

Table 28. 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	23:1	–	–	Reserved.
	0	In	FORCE	Force a device disconnect without SPDM traffic.
7h		In	VERSION	Set to 0.
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.

7.11.2 Actions

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

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, firmware returns INVALID_PAGE_STATE.

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

The firmware checks that this device has no bound or unbound TDIs. If the device has a TDI, 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 set the FORCE flag. When host software either provides a valid SPDM response packet or sets the FORCE flag, 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.

7.11.3 Status Codes

Table 29. 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.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
IN_USE	A TDI is still bound to the device.
SPDM_REQUEST	Indicates that firmware needs to send an SPDM message to the device, and the pending request is in the SPDM request buffer.
SPDM_ERROR	Indicates an error in the SPDM channel.
INCORRECT_BUFFER_LENGTH	Buffer size does not match the buffer size provided by the TIO_STATUS command.
RESYNC_REQUIRED	The SPDM session needs to be resynced. This occurs when the SPDM generates resync required due to some inconsistency observed as defined in the spec.

7.12 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 by 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.

7.12.1 Parameters

Table 30. Layout of the CMD_TIO_DEV_STATUS Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length in bytes of this command buffer.
4h	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
8h	63:0	In	DEV_CTX_PADDR	SPA of a device context page.
10h	63:0	In	STATUS_PADDR	SPA of the DEV_STATUS structure. See *Table 31 below*.
18h-1Fh		–	–	Reserved.

Table 31. Layout of the DEV_STATUS Structure

Byte Offset	Bits	Name	Description
0h	31:0	LENGTH	Length of this structure in bytes.
4h	7:0	DEVICE_STATE	0: Not connected 1: Connected 2: Resync 3: Error 4 to FFh are reserved
5h	7:0	–	Reserved.
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.
Eh-Fh		–	Reserved.
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.

Byte Offset	Bits	Name	Description
	1	NO_FW_UPDATE	Indicates if firmware updates have been 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_NONCE contains the digest of a device attestation report and NONCE value retrieved since TIO_DEV_CONNECT was invoked.
1Dh-1Fh		–	Reserved.
20h-27h		IDE_STREAM_ID[]	Array of 8x 1-byte 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.
28h-2Fh		–	Reserved.
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.
90h	31:0	TDI_CTX_COUNT	Number of TDIs with context buffers associated with this device.
94h	31:0	TDI_CTX_BOUND_COUNT	Number of TDIs bound to guests.
98h-9Fh		SPDM_ALGORITHMS	Algorithms used to establish SPDM session. Refer to the SPDM_ALGOS *Table 75* in this document for list of possible algorithms.

7.12.2 Actions

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

The firmware checks that the STATUS_PADDR is 8-byte aligned and does not cross a 4k page boundary, if not firmware returns INVALID_PARAM. The firmware then checks that the page is in the Firmware or Default page state. If not, 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.

7.12.3 Status Codes

Table 32. 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.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.

7.13 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_DIGEST_FRESH.

MEAS_DIGEST is the digest with NONCE 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_DIGEST_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.

7.13.1 Parameters

Table 33. Layout of Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length in bytes of this command buffer.
4h	23:1	–	–	Reserved.
	0	In	RAW_BITSTREAM	0: Requests the digest form of the attestation report. 1: Requests the raw bitstream form of the attestation report.
7h	7:0	In	VERSION	Set to 0.
8h-27h		In	SPDM_CTRL	SPDM control structure defined in *Section 5.1*.

Byte Offset	Bits	In/Out	Name	Description
28h	63:0	In	DEV_CTX_SLA	Scatter list address of the device context buffer.
30h	255:0	In	MEAS_NONCE	NONCE to be associated with the measurements.

7.13.2 Actions

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

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, firmware returns INVALID_PAGE_STATE.

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

The firmware checks that the device is connected. If not, firmware returns INVALID_STATE.

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.

7.13.3 Status Codes

Table 34. 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.
INVALID_STATE	The device is not connected.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
SPDM_REQUEST	Indicates that firmware needs to send an SPDM message to the device, and the pending request is in the SPDM request buffer.
SPDM_ERROR	Indicates an error in the SPDM channel.
RESYNC_REQUIRED	The SPDM session needs to be resynced. This occurs when the SPDM generates resync required due to some inconsistency observed as defined in the spec.
INCORRECT_BUFFER_LENGTH	Buffer size does not match the buffer size provided by the TIO_STATUS command.

Status	Condition
EXPAND_BUFFER_LENGTH_REQUEST	Indicates that a firmware write buffer is not large enough. Indicates that the firmware is asking to increase the data buffer size. The new buffer size is specified in the BUFFER_CAPACITY field of BUFFER structure.
INVALID_DEV_STATE	The device state is invalid.
RESPONSE_TOO_LARGE	The response buffer is not large enough.

7.14 TIO_DEV_CERTIFICATES

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

7.14.1 Parameters

Table 35. 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	23:1	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
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.

7.14.2 Actions

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

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, firmware returns INVALID_PAGE_STATE.

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

7.14.3 Status Codes

Table 36. 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.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
SPDM_REQUEST	Indicates that firmware needs to send an SPDM message to the device, and the pending request is in the SPDM request buffer.
SPDM_ERROR	Indicates an error in the SPDM channel.
RESYNC_REQUIRED	The SPDM session needs to be resynced. This occurs when the SPDM generates resync required due to some inconsistency observed as defined in the spec.
INCORRECT_BUFFER_LENGTH	Buffer size does not match the buffer size provided by the TIO_STATUS command.
EXPAND_BUFFER_LENGTH_REQUEST	Indicates that a firmware write buffer is not large enough. Indicates that the firmware is asking to increase the data buffer size. The new buffer size is specified in the BUFFER_CAPACITY field of BUFFER structure.
INVALID_DEV_STATE	The device state is invalid.
RESPONSE_TOO_LARGE	The response buffer is not large enough.
DEV_CERT_CHANGED	Digest of the certificate received in GET_CERTIFICATE does not match the digest previously saved digest.

7.15 TIO_DEV_KEY_UPDATE

Update the IDE key used for the device. The command is supported only when TioDevKeyUpdate bit 3 in Fn8000_0024_EBX_x00 is present.

7.15.1 Parameters

Table 37. Layout of the CMD_TIO_DEV_KEY_UPDATE Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length in bytes of this command buffer.
4h	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
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.

7.15.2 Actions

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

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, firmware returns INVALID_PAGE_STATE.

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

If this is the first time TIO_DEV_KEY_UPDATE is invoked for this device, firmware will store relevant input parameters from the device context and the SPDM_CTRL SLAs. For each subsequent call of TIO_DEV_KEY_UPDATE, firmware will verify that these settings have not changed. If it has, then firmware will return INVALID_PARAM.

Firmware will carry out the necessary messaging to update the IDE key used for device traffic, reset the SPDM scratch buffer, and return SUCCESS.

7.15.3 Status Codes

Table 38. Status Codes for TIO_DEV_KEY_UPDATE

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_PLATFORM_STATE	The platform is not in the INIT state.
SPDM_REQUEST	Indicates that firmware needs to send an SPDM message to the device, and the pending request is in the SPDM request buffer.
SPDM_ERROR	Indicates an error in the SPDM channel.
RESYNC_REQUIRED	The SPDM session needs to be resynced. This occurs when the SPDM generates resync required due to some inconsistency observed as defined in the spec.
INCORRECT_BUFFER_LENGTH	Buffer size does not match the buffer size provided by the TIO_STATUS command.
EXPAND_BUFFER_LENGTH_REQUEST	Indicates that a firmware write buffer is not large enough. Indicates that the firmware is asking to increase the data buffer size. The new buffer size is specified in the BUFFER_CAPACITY field of BUFFER structure.
INVALID_DEV_STATE	The device state is invalid.
RESPONSE_TOO_LARGE	The response buffer is not large enough.

7.16 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.

7.16.1 Parameters

Table 39. 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	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
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 buffer to be used as a TDI context page.
18h-23h		In	INTERFACE_ID	Interface ID of the TDI as defined by TDISP.
24h-2Fh		–	–	Reserved.

7.16.2 Actions

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

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

The firmware checks that the device is connected. If not, firmware returns INVALID_DEV_STATE.

The firmware checks that TDI_CTX_SLA is a valid scatter list address. If not, 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, 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, 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, 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.

7.16.3 Status Codes

Table 40. 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.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
IN_USE	The TDI is already in use.
INVALID_DEV_STATE	The device is not connected.

7.17 TIO_TDI_RECLAIM

Reclaim a TDI context page.

This command does not generate SPDM traffic with the device.

7.17.1 Parameters

Table 41. 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	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
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.
18h-1Fh		–	–	Reserved.

7.17.2 Actions

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

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

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

The firmware transitions the page to the Reclaim page state.

7.17.3 Status Codes

Table 42. 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.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
IN_USE	The TDI is already bound to a guest.
INVALID_DEV_STATE	The device state is invalid.

7.18 TIO_TDI_BIND

Bind a TDI to a guest.

Host software uses this command to bind a guest and a TDI together.

The TIO_TDI_BIND command provides support for transitioning the TDI state via two different methods:

1. Transition TDI (CONFIG_LOCKED or CONFIG_UNLOCKED) directly to the RUN state.
2. Transition TDI (CONFIG_UNLOCKED or RUN) state to CONFIG_LOCKED state.

There are a set of command transition definitions to support these three transition actions.

Table 43. Command Transition Definitions

Current State	State Transition Bit (STATE_TRANSITION)	Transition Requested	SPDM Output Buffer
CONFIG_UNLOCKED	1	CONFIG_UNLOCKED to RUN	Yes

Current State	State Transition Bit (STATE_TRANSITION)	Transition Requested	SPDM Output Buffer
CONFIG_UNLOCKED	0	CONFIG_UNLOCKED to CONFIG_LOCKED	Yes
CONFIG_LOCKED	1	CONFIG_LOCKED to RUN	Yes
CONFIG_LOCKED	0	CONFIG_LOCKED to RUN	No

In the CONFIG_UNLOCKED state, the TIO_TDI_BIND command with STATE_TRANSITION set to 1 is a command to transition the TDI state from CONFIG_UNLOCKED to the RUN state, transitioning through the CONFIG_LOCKED state.

In the CONFIG_UNLOCKED state, the TIO_TDI_BIND command with STATE_TRANSITION set to 0 is a command to transition the TDI state from CONFIG_UNLOCKED to the CONFIG_LOCKED state.

***Note**: Supporting transitioning into the persistent CONFIG_LOCKED state provides an opportunity for the Host software to facilitate providing additional configuration elements to the TDI.*

In the CONFIG_LOCKED state, the TIO_TDI_BIND command with STATE_TRANSITION set to 1 is a command to transition the TDI state from CONFIG_LOCKED to the CONFIG_RUN state.

In the CONFIG_LOCKED state, the TIO_TDI_BIND command with STATE_TRANSITION set to 0 is an invalid command and firmware returns INVALID_COMMAND.

After successful transition to the RUN state, the guest can assess the authenticity and configuration of the device to determine whether the guest wants to use the TDI.

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

7.18.1 Parameters

Table 44. Layout of the CMD_TIO_TDI_BIND Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length in bytes of this command buffer.
4h	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Must be set to 0x01.
8h-27h		In	SPDM_CTRL	SPDM control structure defined in *Section 5.1*.

Byte Offset	Bits	In/Out	Name	Description
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	In	GUEST_DEVICE_ID	Device ID assigned by hypervisor to guest.
42h	15:5	–	–	Reserved. Must be 0.
	4	–	AL_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	15:1	–	–	Reserved. Must be 0.
	0	In	STATE_TRANSITION	0: Drive TDI to CONFIG_LOCKED state 1: Drive TDI to RUN state
46h	15:0	In	QUEUE_ID	Identifier used by IOMMU for TDI I/O memory mapping.
48h	15:0	In	DOMAIN_ID	Host-assigned PCIe domain for device associated with TDI.
4Ah	47:0	–	–	Reserved. Must be 0.
50h	63:0	In	MMIO_REPORTING_OFFSET	The offset added to physical address (in bytes) of MMIO base in device interface report.
58h-5Fh		–	–	Reserved. Must be 0.

7.18.2 Version History

Version	Description	Version of SEV-TIO ABI
00h	Initial.	0.91
01h	Added DOMAIN_ID (device), QUEUE_ID (IOMMU), MMO_REPORTING_OFFSET and VERSION.	0.92

7.18.3 Actions

7.18.3.1 Initial TDI State Command Transition Validation

If the TDI state is CONFIG_LOCKED and STATE_TRANSITION is set to 0, firmware returns INVALID_STATE.

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

The firmware checks that the MMIO_REPORTING_OFFSET is aligned to a 4 KB boundary. If not, firmware returns INVALID_ADDRESS.

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

Firmware checks that the TDI is not bound to a guest. If the TDI is bound to a guest, firmware returns IN_USE. Otherwise, firmware will store the DOMAIN_ID and QUEUE_ID in the TDI context.

7.18.3.2 CONFIG_LOCKED State

If the current state is CONFIG_LOCKED, then the following actions are performed:

The firmware transitions the TDI state to the RUN state, passing the provided flags to the LOCK_INTERFACE_REQUEST message.

The firmware retrieves the interface report of the TDI and then calculates and saves the digest of the interface report in the TDI context buffer. The firmware increments the TDIReportCount value in the TDI context.

The TIO_TDI_BIND command completes.

7.18.3.3 CONFIG_UNLOCKED State

If the current state is CONFIG_UNLOCKED then the following actions are performed.

If the STATE_TRANSITION bit is unset, the firmware transitions the TDI to the CONFIG_LOCKED state, passing the provided flags to the LOCK_INTERFACE_REQUEST message.

If the STATE_TRANSITION bit is set, then transition the TDI state to the RUN state passing the provided flags to the LOCK_INTERFACE_REQUEST message.

The firmware retrieves the interface report of the TDI and then calculates and saves the digest of the interface report in the TDI context buffer. The firmware sets the TDIReportCount value in the TDI context to 1.

The TIO_TDI_BIND command completes.

7.18.4 Status Codes

Table 45. 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.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
IN_USE	The guest device ID is already in use.
INVALID_STATE	Invalid state transition request.
SPDM_REQUEST	Indicates that firmware needs to send an SPDM message to the device, and the pending request is in the SPDM request buffer.
SPDM_ERROR	Indicates an error in the SPDM channel.
RESYNC_REQUIRED	The SPDM session needs to be resynced. This occurs when the SPDM generates resync required due to some inconsistency observed as defined in the spec.
INCORRECT_BUFFER_LENGTH	Buffer size does not match the buffer size provided by the TIO_STATUS command.
EXPAND_BUFFER_LENGTH_REQUEST	Indicates that a firmware write buffer is not large enough. Indicates that the firmware is asking to increase the data buffer size. The new buffer size is specified in the BUFFER_CAPACITY field of BUFFER structure.
RESPONSE_TOO_LARGE	The response buffer is not large enough.

7.19 TIO_TDI_UNBIND

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

7.19.1 Parameters

Table 46. 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	23:1	–	–	Reserved.

Byte Offset	Bits	In/Out	Name	Description
	0	In	FORCE	Indicates if device should be forcefully unbound regardless of the SPDM status.
7h	7:0	In	VERSION	Set to 0.
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.

7.19.2 Actions

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

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

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

Firmware checks if the FORCE bit is set to 1. If it is then it unbinds the TDI regardless of the SPDM state and releases TDI resources.

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.

The firmware sets the TDIReportCount value in the TDI context to 0.

The TIO_TDI_UNBIND command completes successfully.

7.19.3 Status Codes

Table 47. 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.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_GUEST	The guest is invalid.
SPDM_REQUEST	Indicates that firmware needs to send an SPDM message to the device, and the pending request is in the SPDM request buffer.
SPDM_ERROR	Indicates an error in the SPDM channel.
RESYNC_REQUIRED	The SPDM session needs to be resynced. This occurs when the SPDM generates resync required due to some inconsistency observed as defined in the spec.
EXPAND_BUFFER_LENGTH_REQUEST	Indicates that a firmware write buffer is not large enough. Indicates that the firmware is asking to increase the data buffer size. The new buffer size is specified in the BUFFER_CAPACITY field of BUFFER structure.
INVALID_TDI	Indicates that the identified TDI is invalid.

7.20 TIO_TDI_REPORT

Retrieve the TDI interface report.

7.20.1 Parameters

Table 48. 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	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
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.

7.20.2 Actions

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

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

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

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

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, firmware returns INVALID_PAGE_STATE.

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

The firmware retrieves the interface report of the TDI. 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. The firmware increments the TDIReportCount value in the TDI context.

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

7.20.3 Status Codes

Table 49. 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.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
SPDM_REQUEST	Indicates that firmware needs to send an SPDM message to the device, and the pending request is in the SPDM request buffer.
SPDM_ERROR	Indicates an error in the SPDM channel.

Status	Condition
RESYNC_REQUIRED	The SPDM session needs to be resynced. This occurs when the SPDM generates resync required due to some inconsistency observed as defined in the spec.
INCORRECT_BUFFER_LENGTH	Buffer size does not match the buffer size provided by the TIO_STATUS command.
EXPAND_BUFFER_LENGTH_REQUEST	Indicates that a firmware write buffer is not large enough. Indicates that the firmware is asking to increase the data buffer size. The new buffer size is specified in the BUFFER_CAPACITY field of BUFFER structure.
RESPONSE_TOO_LARGE	The response buffer is not large enough.
INVALID_DEV_STATE	The device state is invalid.
INVALID_TDI_STATE	The TDI state is invalid.
INVALID_DEV_CTX	Indicates that the provided page is not a valid device context page.
INVALID_TDI_CTX	Indicates that the provided page is not a valid TDI context page.

7.21 TIO_TDI_INFO

Retrieve information about the TDI status.

This command does not generate SPDM traffic with the device.

7.21.1 Parameters

Table 50. 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	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Must be set to 0x01.
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.
18h	63:0	In	TDI_INFO_PADDR	System physical address of TDI_INFO structure. See *Table 51 below*.
20h-2Fh		–	–	Reserved.

Table 51. 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	31:4	–	Reserved.
	3:2	TDI_STATUS	0: TDI_UNBOUND 1: TDI_BIND_LOCKED 2: TDI_BIND_RUN

Byte Offset	Bits	Name	Description
	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.
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 75* 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	63:0	TDI_REPORT_COUNT	TDIReportCount value in the TDI context.
B8h	31:0	ASID	ASID of the guest that this device is assigned to. Valid only if CTX_STATE is 1.
BCh	31:0	–	Reserved.
C0h	63:0	TDI_ID	SNP-assigned identifier for the TDI

7.21.2 Version History

Version	Description	Version of SEV-TIO ABI
00h	Initial version.	0.91
01h	TDI_ID returned in TDI_INFO.	0.92

7.21.3 Actions

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

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

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

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

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

The firmware fills the TDI_INFO buffer with status information.

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.

7.21.4 Status Codes

Table 52. 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.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_DEV_STATE	The device state is invalid.
INVALID_TDI_STATE	The TDI state is invalid.
INVALID_DEV_CTX	Indicates that the provided page is not a valid device context page.
INVALID_TDI_CTX	Indicates that the provided page is not a valid TDI context page.

7.22 TIO_TDI_STATUS

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

7.22.1 Parameters

Table 53. 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	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
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.
28h	63:0	In	STATUS_PADDR	System physical address of TDI_STATUS structure. See *Table 51.*

Table 54. Layout of the TIO_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-Fh		–	Reserved.

7.22.2 Actions

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

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

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

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

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, firmware returns INVALID_PAGE_STATE.

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

The firmware fills the TDI_STATUS buffer with status information retrieved from the device.

7.22.3 Status Codes

Table 55. 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.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
SPDM_REQUEST	Indicates that firmware needs to send an SPDM message to the device, and the pending request is in the SPDM request buffer.
SPDM_ERROR	Indicates an error in the SPDM channel.
RESYNC_REQUIRED	The SPDM session needs to be resynced. This occurs when the SPDM generates resync required due to some inconsistency observed as defined in the spec.
INCORRECT_BUFFER_LENGTH	Buffer size does not match the buffer size provided by the TIO_STATUS command.
EXPAND_BUFFER_LENGTH_REQUEST	Indicates that a firmware write buffer is not large enough. Indicates that the firmware is asking to increase the data buffer size. The new buffer size is specified in the BUFFER_CAPACITY field of BUFFER structure.
INVALID_TDI_STATE	The TDI state is invalid.
RESPONSE_TOO_LARGE	The response buffer is not large enough.
INVALID_DEV_STATE	The device state is invalid.
INVALID_DEV_CTX	Indicates that the provided page is not a valid device context page.
INVALID_TDI_CTX	Indicates that the provided page is not a valid TDI context page.

7.23 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.

7.23.1 Parameters

Table 56. 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	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
8h	63:0	In	DEV_CTX_SLA	Scatter list address of the device context buffer.
10h	63:0	In	GCTX_PADDR	System physical address of a guest context page.
18h-1Fh		–	–	Reserved.

7.23.2 Actions

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

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

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

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

7.23.3 Status Codes

Table 57. 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_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_GUEST	The guest is invalid (the provided guest context is invalid).
IN_USE	The resource is in use.
INVALID_DEV_STATE	The device state is invalid.
INVALID_DEV_CTX	Indicates that the provided page is not a valid device context page.

7.24 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.

7.24.1 Parameters

Table 58. Layout of the 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	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
8h	63:0	In	DEV_CTX_SLA	Scatter list address of the device context buffer.
Ch	31:0	In	ASID	ASID of a guest to retrieve status.
14h	63:0	In	STATUS_PADDR	SPA of an aligned 8B location to write the status. Bits 1:0 DMA fence status 2'b00: ASID is not fenced 2'b01: ASID is DMA fenced due to error 2'b02: reserved 2'b03: ASID is DMA fenced by default Bit 3:2 MMIO fence status 2'b00: ASID is not fenced 2'b01: Reserved 2'b02: Reserved 2'b03: ASID is MMIO fenced.
1Ch-1Fh		–	–	Reserved.

7.24.2 Actions

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

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

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

7.24.3 Status Codes

Table 59. Status Codes for TIO_ASID_FENCE_STATUS

Status	Condition
SUCCESS	Successful completion.
INVALID_ADDRESS	A provided address was invalid.

Status	Condition
INVALID_PAGE_STATE	A provided page was not in the expected page state.
INVALID_PLATFORM_STATE	The platform is not in the INIT state.
INVALID_DEV_STATE	The device state is invalid.
INVALID_DEV_CTX	Indicates that the provided page is not a valid device context page.

7.25 TIO_VIOMMU_INIT

This command is used to allocate memory for MMIO backing pages in an IOMMU instance to support VIOMMU functionality. These pages are brought into the IOMMU private memory. 8MB of memory is required for each IOMMU instance. The IOMMU instance is addressed using its PCIe identifier.

7.25.1 Parameters

Table 60. Layout of the TIO_VIOMMU_INIT Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length in bytes of this command buffer.
4h	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
8h	15:0	In	SEGMENT_ID	Segment ID corresponding to target IOMMU.
Ah	15:0	In	IOMMU_DEVICE_ID	IOMMU unique PCIe identifier using Bus:Device::Function (BDF) format. Function is encoded in 4 LSBs, with lead bit set to 0 (i.e., 8 possible functions).
Ch	63:0	In	VIOMMU_PADDR	Physical address of 8 MB of contiguous backing pages for MMIO space, to be used by IOMMU for VIOMMU functionality.
14h-1Fh		–	–	Reserved.

7.25.2 Actions

The firmware checks that SNP_INIT_EX was invoked with the TIO_EN and VIOMMU_EN flags set. If not, firmware returns INVALID_PLATFORM_STATE.

The firmware confirms that the sPA given in VIOMMU_PADDR is valid and points to a contiguous 8 MB memory region. If not, it returns INVALID_ADDRESS.

The firmware then checks that the pages pointed at by VIOMMU_PADDR are a Firmware or Default page. If not, firmware returns INVALID_PAGE_STATE.

Firmware will verify that SEGMENT_ID is in the allowed PCIe bus range and IOMMU_DEVICE_ID is in the valid BDF range. If not, it will return INVALID_PARAM.

Firmware will then convert the pages to IOMMU private pages.

7.25.3 Status Codes

Table 61. Status Codes for TIO_VIOMMU_INIT

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	SNP_INIT_EX was not invoked with TIO_EN and VIOMMU_EN set.
INVALID_ADDRESS	VIOMMU_PADDR was invalid.
INVALID_PARAM	IOMMU addressing is not valid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.

7.26 TIO_VIOMMU_GUEST_INIT

This command is used by the hypervisor to initialize the VIOMMU feature for a guest.

7.26.1 Parameters

Table 62. Layout of the TIO_VIOMMU_GUEST_INIT Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length in bytes of this command buffer.
4h	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
8h	63:0	In	GCTX_PADDR	System physical address of a guest context page.
10h	15:0	In	SEGMENT_ID	Segment ID corresponding to target IOMMU.
12h	15:0	In	IOMMU_DEVICE_ID	IOMMU unique PCIe identifier using Bus:Device::Function (BDF) format. Function is encoded in 4 LSBs, with lead bit set to 0 (i.e. 8 possible functions).
14h	15:0	In	GUEST_VIOMMU _ID	Unique (per guest) identifier for VIOMMU
16h	15:0	In	HOST_VIOMMU_ID	Host ID used by IOMMU for GPA to SPA translation. This mapping is required for IOMMU to access guest memory for command, event log and ppr buffers.
18h	15:0	In	DOMAIN_ID	Host domain ID for HOST_VIOMMU_ID.
1Ah	47:0	–	–	Reserved.
20h	63:0	In	DEVICE_TABLE_PADDR	System physical address of Device Mapping Table used for guest. This must point to 1 MB of contiguous memory.

Byte Offset	Bits	In/Out	Name	Description
28h	63:0	In	DOMAIN_TABLE_PADDR	System physical address of Domain Mapping Table used for guest. This must point to 524,.288 kbytesB (i.e., 80000h bytes) of contiguous memory.
30h	63:0	In	GUEST_MMIO_PADDR	System physical address of the guest virtual function MMIO page. This page must be 4 kB-aligned.
38h-3Fh		–	–	Reserved.

7.26.2 Actions

The firmware checks that SNP_INIT_EX was invoked with the TIO_EN and VIOMMU_EN flags set. If not, firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR is a Context page. If not, firmware returns INVALID_GUEST.

Firmware will verify that SEGMENT_ID is in the allowed PCIe bus range and IOMMU_DEVICE_ID is in the valid BDF range. If not, it will return INVALID_PARAM.

The firmware checks that the address in DEVICE_TABLE_PADDR is a valid sPA and all 1 MB of contiguous memory starting from that address is addressable. If not, it returns INVALID_ADDRESS. The firmware then checks that the pages pointed at by DEVICE_TABLE_PADDR are Firmware or Default pages. If not, firmware returns INVALID_PAGE_STATE.

The firmware checks that the address in DOMAIN_TABLE_PADDR is a valid sPA, and all 512 KB of contiguous memory starting from that address is addressable. If not, it returns INVALID_ADDRESS. The firmware then checks that the pages pointed at by DOMAIN_TABLE_PADDR are Firmware or Default pages. If not, firmware returns INVALID_PAGE_STATE.

The firmware checks that the address in GUEST_MMIO_PADDR is a valid sPA. If not, it returns INVALID_ADDRESS. The firmware then checks that the page pointed at by GUEST_MMIO_PADDR is a Firmware or Default page. If not, firmware returns INVALID_PAGE_STATE.

Firmware will determine if there is an existing VIOMMU instance for the guest on the target IOMMU. If there is, firmware will return IN_USE. If not, firmware will add the SEGMENT_ID, IOMMU_DEVICE_ID, GUEST_VIOMMU_ID, HOST_VIOMMU_ID and DOMAIN_ID as non-exportable guest context fields (SegmentId, IommuDeviceId, GuestViommuId, HostViommuId and DomainId).

Firmware will convert the pages pointed to by DEVICE_TABLE_PADDR, DOMAIN_TABLE_PADDR and GUEST_MMIO_PADDR to IOMMU private pages. Firmware will overwrite the corresponding GPAs in the RMP with IOMMU Private Addresses (IPAs).

7.26.3 Status Codes

Table 63. Status Codes for TIO_VIOMMU_GUEST_INIT

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	SNP_INIT_EX was not invoked with TIO_EN and VIOMMU_EN set.
INVALID_ADDRESS	VIOMMU_PADDR was invalid.
INVALID_PARAM	IOMMU addressing is not valid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
IN_USE	The target IOMMU already has a VIOMMU for the guest.

7.27 TIO_VIOMMU_GUEST_SHUTDOWN

This command is used by the hypervisor to terminate a VIOMMU instance for the guest.

7.27.1 Parameters

Table 64. Layout of the TIO_VIOMMU_GUEST_SHUTDOWN Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length in bytes of this command buffer.
4h	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
8h	63:0	In	GCTX_PADDR	System physical address of a guest context page.
10h	15:0	In	SEGMENT_ID	Segment ID corresponding to target IOMMU.
12h	15:0	In	IOMMU_DEVICE_ID	IOMMU unique PCIe identifier using Bus:Device::Function (BDF) format. Function is encoded in 4 LSBs, with lead bit set to 0 (i.e., 8 possible functions).
14h-1Fh		–	–	Reserved.

7.27.2 Actions

The firmware checks that SNP_INIT_EX was invoked with the TIO_EN and VIOMMU_EN flags set. If not, firmware returns INVALID_PLATFORM_STATE.

The firmware checks that GCTX_PADDR is a Context page. If not, firmware returns INVALID_GUEST.

Firmware will verify that SEGMENT_ID is in the allowed PCIe bus range and IOMMU_DEVICE_ID is in the valid BDF range. If not, it will return INVALID_PARAM.

Firmware will determine if there is an existing VIOMMU instance for the guest on the target IOMMU. If there is not, firmware will return SUCCESS. If there is an instance, firmware will remove the corresponding guest context fields (SegmentId, IommuDeviceId, GuestViommuId, HostViommuId and DomainId).

Firmware will invalidate the guest MMIO page and reflect that in the RMP. Firmware will restore all IOMMU private pages to reclaimable status.

Firmware will reset the VIOMMU of the guest on the target IOMMU. If for any reason this operation fails, firmware will return HARDWARE_PLATFORM. Otherwise it will return SUCCESS.

7.27.3 Status Codes

Table 65. Status Codes for TIO_VIOMMU_GUEST_SHUTDOWN

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	SNP_INIT_EX was not invoked with TIO_EN and VIOMMU_EN set.
INVALID_ADDRESS	VIOMMU_PADDR was invalid.
INVALID_PARAM	IOMMU addressing is not valid.
INVALID_PAGE_STATE	A provided page was not in the expected page state.
HARDWARE_PLATFORM	IOMMU failed to reset VIOMMU instance.

7.28 TIO_VIOMMU_STATUS

This command is used to retrieve the VIOMMU status.

7.28.1 Parameters

Table 66. Layout of the TIO_VIOMMU_STATUS Structure

Byte Offset	Bits	In/Out	Name	Description
0h	31:0	In	LENGTH	Length in bytes of this command buffer.
4h	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
8h	63:0	In	VIOMMU_STATUS_PADDR	System physical address where the TIO_VIOMMU_STATUS data structure is returned.

The firmware checks that SNP_INIT_EX was invoked with the TIO_EN and VIOMMU_EN flags set. If not, firmware returns INVALID_PLATFORM_STATE.

The firmware checks that the VIOMMU_STATUS_PADDR is 8-byte aligned and not cross a 4k page boundary, if not firmware returns INVALID_ADDRESS The firmware then checks that the page is in the Firmware or Default page state. If not, firmware returns INVALID_PAGE_STATE

Otherwise firmware will write the VIOMMU_STATUS structure (see *Table 67 below*) at the address indicated by VIOMMU_STATUS_PADDR.

7.28.2 Actions

Table 67. VIOMMU_STATUS Return Structure

Byte Offset	Bits	Name	Description
0h	0:31	LENGTH	VIOMMU status data length in bytes.
4h	0:7	VERSION	Set to 1h for this revision.
5h	0:23	–	Reserved.
8h	0:15	NUM_IOMMU	Number of IOMMU in system.
Ah	0:47	Reserved	Reserved. Mbz.
10h	0:383	vIOMMU Status Block	VIOMMU status for each block (see *Table 68*Table 68. VIOMMU_STATUS_BLOCK Return Structure).

Table 68. VIOMMU_STATUS_BLOCK Return Structure

Byte Offset	Bits	Name	Description
0h	0:15	IOMMU_SEGMENT_ID	PCIe segment ID for IOMMU
2h	0:31	IOMMU_DEVICE_ID	PCIe device ID for IOMMU.
4h	0:31	FLAGS	Bit 0: 1 if VIOMMU_INIT is done. 0 otherwise. All other bits are 0.
8h	0:63	VIOMMU_MMIO_1_NS_PADDR	SPA of VIOMMU MMIO Region 1 backing storage for non-secure guests.
10h	0:63	VIOMMU_MMIO_1_NS_PADDR	SPA of VIOMMU MMIO Region 1 backing storage for secure guests.
18h	0:63	VIOMMU_MMIO_2_PADDR	SPA of VIOMMU MMIO Region 2 backing storage.
20h	0:15	NUM_SECURE_GUESTS	Number of secure guests using VIOMMU.
22h	0:111	Reserved	Reserved. Mbz.

7.28.3 Status Codes

Table 69. Status Codes for TIO_VIOMMU_STATUS

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	SNP_INIT_EX was not invoked with TIO_EN and VIOMMU_EN set.
INVALID_ADDRESS	VIOMMU_PADDR was invalid.
INVALID_PARAM	Reserved bits are not set to zero.

Status	Condition
INVALID_PAGE_STATE	A provided page was not in the expected page state.
HARDWARE_PLATFORM	IOMMU failed to reset VIOMMU instance.

7.29 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.

7.29.1 Parameters

Table 70. 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	23:0	–	–	Reserved.
7h	7:0	In	VERSION	Set to 0.
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.

7.29.2 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.

The GCTX_PADDR and GUEST_RESPONSE_PADDR must be aligned on a 4k byte boundary, and the size of the GUEST_RESPONSE_PADDR buffer must not cross a 4k boundary. The firmware then checks that the page pointed at by the GUEST_RESPONSE_PADDR is a Firmware or a Default page. If not, firmware returns INVALID_PAGE_STATE.

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 71*.

Table 71. Message Type Encodings

Default	Message Type	Message Version
0	Invalid	-
19	TIO_MSG_TDI_INFO_REQ	2
20	TIO_MSG_TDI_INFO_RSP	2
21	TIO_MSG_MMIO_VALIDATE _REQ	2
22	TIO_MSG_ MMIO_VALIDATE _RSP	2
23	TIO_MSG_MMIO_CONFIG_REQ	2
24	TIO_MSG_MMIO_CONFIG_RSP	2
25	TIO_MSG_ SDTE_WRITE_REQ	2
26	TIO_MSG_SDTE_WRITE_RSP	2
27	TIO_MSG_CONFIG_VIOMMU_REQ	1
28	TIO_MSG_CONFIG_VIOMMU_RSP	1
29	TIO_MSG_CONFIG_VIOMMU_MAPPING_REQ	1
30	TIO_MSG_CONFIG_VIOMMU_MAPPING_RSP	1

The semantics of each guest message are specified *Chapter 8*.

7.29.3 Status Codes

Table 72. Status Codes for TIO_GUEST_REQUEST

Status	Condition
SUCCESS	Successful completion.
INVALID_PLATFORM_STATE	SNP_INIT_EX was not invoked with TIO_EN and VIOMMU_EN set.
INVALID_PARAM	Reserved bits are not set to zero.
INVALID_ADDRESS	VIOMMU_PADDR was invalid.
INVALID_GUEST	The guest is invalid (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.

Status	Condition
AEAD_OFLOW	The message sequence number was incorrect, or the guest's message count would overflow.
BAD_MEASUREMENT	The message failed to authenticate.
INVALID_DEV_STATE	The device state is invalid.
INVALID_DEV_CTX	Indicates that the provided page is not a valid device context page.

Chapter 8 SEV-TIO Guest Messages

8.1 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 73. Layout of the TIO_MSG_TDI_INFO_REQ Structure

Byte Offset	Bits	Name	Description
0h	63:0	TDI_ID	Firmware provided identifier used by the guest to identify the TDI in guest messages.
8h-Fh		–	Reserved.

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

Table 74. Layout of the TIO_MSG_TDI_INFO_RSP Structure

Byte Offset	Bits	Name	Description
0h	63:0	TDI_ID	Firmware provided identifier used by the guest to identify the TDI in guest messages.
8h	15:0	TDI_STATUS	0: Success (Bound TDI) 1: Invalid TDI 2: Unbound TDI All other encoding is reserved.
Ah-Fh		–	Reserved.
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	31:5	–	Reserved.

Byte Offset	Bits	Name	Description
	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.
18h-1Fh		SPDM_ALGOS	Algorithms used to establish the SPDM session. See *Table 75* 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 DI.
80h	383:0	INTERFACE_REPORT_DIGEST	Digest of interface report.
B0h	63:0	TDI_REPORT_COUNT	TDIReportCount value in the TDI context.
B8h-BFh	63:0	–	Reserved.

Table 75. Layout of the SPDM_ALGOS Structure

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

8.2 MMIO Validate

This message asks the SEV firmware to set 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 76. Layout of the TIO_MSG_MMIO_VALIDATE_REQ Structure

Byte Offset	Bits	Name	Description
0h	63:0	TDI_ID	Firmware provided identifier used by the guest to identify the TDI in guest messages.
8h-Fh		–	Reserved.
10h	63:0	SUBRANGE_BASE	Guest physical address of the subrange.
18h	31:0	SUBRANGE_PAGE_COUNT	Number of 4 KB pages in the subrange.
1Ch	31:0	RANGE_OFFSET	Offset (in 4 KB pages) of the subrange within the MMIO range.
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-2Fh		–	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, firmware returns the appropriate error status code.

The firmware sets VALIDATED in 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 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 77. Layout of the TIO_MSG_MMIO_VALIDATE_RSP Structure

Byte Offset	Bits	Name	Description
0h	63:0	TDI_ID	Firmware provided identifier used by the guest to identify the TDI in guest messages.
8h	15:0	STATUS	0: Success

Byte Offset	Bits	Name	Description
			1: Invalid TDI 2: TDI is not bound to this guest. 3: At least one page is not assigned to the guest. 4: At least one page is not an I/O page. 5: The Validated bit is not uniformly set for the MMIO subrange. 6: At least one page does not have immutable bit set when validated bit is clear 7: At least one page is not mapped to the expected MMIO pages 8: The provided MMIO range ID is not reported in the interface report 9: The subrange is not in the MMIO range in the interface report. 10: At least one page is not 4K page size. *All other encodings reserved.*
Ah-Fh		–	Reserved.
10h	63:0	SUBRANGE_BASE	Guest physical address of the subrange.
18h	31:0	SUBRANGE_BASE_COUNT	Number of 4 KB pages in the subrange.
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-2Fh		–	Reserved.

8.3 VIOMMU Configure

This message requests the SEV firmware to program the VIOMMU MMIO control registers. This message must be sent using SNP_GUEST_REQUEST, not TIO_GUEST_REQUEST.

Table 78. Layout of the SNP_MSG_CONFIG_VIOMMU_REQ Structure

Byte Offset	Bits	Name	Description
0h	15:0	GUEST_VIOMMU_ID	Guest VIOMMU identifier as assigned by the hypervisor.
2h	15:0	–	Reserved.
4h	31:0	OPERATION_FLAGS	Bit 6: Write sDTE for target VIOMMU Bit 5: Set virtual MMIO GPA for the Guest Bit 4: Write Event Log B Control Register Bit 3 Write PPR Log B Control Register Bit 2: Write PPR Control Register Bit 1: Write Event Control Register Bit 0: Write Command Control Register
8h	63:0	GUEST_COMMAND_CONTROL	Valid if Bit 0 is set in OPERATION_FLAGS.

Byte Offset	Bits	Name	Description
10h	63:0	GUEST_EVENT_CONTROL	Valid if Bit 1 is set in OPERATION_FLAGS.
18h	63:0	GUEST_PPR_CONTROL	Valid if Bit 2 is set in OPERATION_FLAGS.
20h	63:0	GUEST_PPR_LOGB_CONTROL	Valid if Bit 3 is set in OPERATION_FLAGS.
28h	63:0	GUEST_EVENT_LOGB_CONTROL	Valid if Bit 4 is set in OPERATION_FLAGS.
30h	63:52	–	Reserved.
	51:12	VF_MMIO_GPA_ADDR	The GPA assigned to the guest. Valid if Bit 5 is set in OPERATION_FLAGS.
	11:1	–	Reserved.
	0	VF_MMIO_GPA_ADDR_VALID	The Validate bit setting.
38h	63:0	–	Reserved.
40h	63:52	–	Reserved.
	51:12	VTOM_ADDR	vTOM GPA. Valid if Bit 6 is set in OPERATION_FLAGS.
	11:1	–	Reserved.
	0	VTOM_EN	vTOM is enabled.
48h	7:2	–	Reserved.
	1:0	VIOMMU_VMPL	VMPL level. Valid if Bit 6 is set in OPERATION_FLAGS.
49h	7:1	–	Reserved.
	0	SDTE_VALID	The valid bit. Valid if Bit 6 is set in OPERATION_FLAGS.
50h-5Fh		–	Reserved.

The firmware verifies that VIOMMU_EN and TIO_EN has been set in SNP_INIT_EX options, and TIO has been initialized. If not, it returns INVALID_PLATFORM_STATE.

Firmware will verify that the caller guest is VMPL0. If not, firmware will return INVALID_GUEST.

Firmware will verify that a VIOMMU corresponding to GUEST_VIOMMU_ID has been configured for the guest. If not, firmware will return INVALID_GUEST.

Firmware will verify that VF_MMIO_GPA_ADDR is a valid GPA assigned to the guest. If not, it will return the invalid address status code.

Firmware will process each of the bits in the OPERATION_FLAGS according to their definition in *Table 78*.

Bits 0 through 4 require configuration of corresponding guest MMIO registers in the IOMMU configuration space if they are set. Bit 5 requires update of the RMP for the backing guest MMIO page (including validation if the option is set). Bit 6 manages the VIOMMU secure device table, including management of regions for the DMA regions.

The firmware responds with the SNP_MSG_CONFIG_VIOMMU_RSP message.

Table 79. Layout of the SNP_MSG_CONFIG_VIOMMU_RSP Structure

Byte Offset	Bits	Name	Description
0h	15:0	GUEST_VIOMMU_ID	Guest VIOMMU identifier as assigned by the hypervisor.
8h	15:0	STATUS	Lower byte: 0: Success 1: Invalid GUEST_VIOMMU_ID 2: Invalid VF_MMIO_GPA Address *All other values reserved.* Upper byte: Bit 6: 1 if sDTE write failed. Bit 5: 1 if setting VF MMIO in RMP failed. Bit 4: 1 if Log B Control Register write failed. Bit 3: 1 if PPR Log B Control Register write failed. Bit 2: 1 if PPR Control Register write failed. Bit 1: 1 if Event Control Register write failed. Bit 0: 1 if Command Control Register write failed. *All other encodings reserved.*
Ah-Fh		–	Reserved.

8.4 VIOMMU MAPPING Configure

This message requests the SEV firmware to configure the domain and/or device identifier mappings for the guest VIOMMU instance.

Table 80. Layout of the TIO_MSG_CONFIG_VIOMMU_MAPPING_REQ Structure

Byte Offset	Bits	Name	Description
0h	63:0	TDI_ID	Firmware provided identifier used by the guest to identify the TDI in guest messages.
8h	15:0	GUEST_VIOMMU_ID	Guest VIOMMU identifier as assigned by the hypervisor.
Ah	7:1	–	Reserved. Must be zero.
	0	OPERATION	If 1 then the operation is to set the mapping value and Valid bit in IOMMU. If 0 then the operation is to clear the mapping value and Valid bit.
Bh	7:2	–	Reserved. Must be zero.
	1:0	FLAGS	Bit 1: Configure domain mapping. Set to 1 when firmware is to write the mapping value in GUEST_DOMAIN_ID to the VIOMMU settings and enable. Otherwise, firmware will clear the current VIOMMU domain ID and invalidate it.

Byte Offset	Bits	Name	Description
			Bit 0: Configure device mapping. Set to 1 when firmware is to write the mapping value in GUEST_DEVICE_ID to the VIOMMU settings and enable. Otherwise firmware will clear the current VIOMMU domain ID and invalidate it.
Ch	15:0	GUEST_DOMAIN_ID	Domain ID to be used by VIOMMU if Bit 1 is set to 1 in FLAGS.
Eh	15:0	–	Reserved. Mbz.

The firmware verifies that TIO_EN and VIOMMU_EN has been set in SNP_INIT_EX options. If not, it returns INVALID_PLATFORM_STATE.

Firmware will verify that a VIOMMU corresponding to GUEST_VIOMMU_ID has been configured for the guest. If not, firmware will return INVALID_GUEST.

Firmware will verify that the TDI_ID refers to a bound TDI assigned to the guest. If not, firmware will return INVALID_TDI.

Firmware will process each of the bits in the OPERATION and FLAGS according to their definition in *Table 80*.

The firmware responds with the TIO_MSG_CONFIG_VIOMMU_MAPPING_RSP message.

Table 81. Layout of the TIO_MSG_CONFIG_VIOMMU_MAPPING_RSP Structure

Byte Offset	Bits	Name	Description
0h	63:0	TDI_ID	Firmware provided identifier used by the guest to identify the TDI in guest messages.
8h	15:0	STATUS	Lower byte: 0: Success 1: Invalid TDI_ID 2: Unbound TDI 3: Operation Failure 4: Hardwar Error *All other values reserved.* Upper byte: Bit 0: 1 if GUEST_DEVICE_ID write failed. Bit 1: 1 if GUEST_DOMAIN_ID write failed. *All other encodings reserved.*
Ah-Fh		–	Reserved.

8.5 MMIO Configure

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

Table 82. Layout of the TIO_MSG_MMIO_CONFIG_REQ Structure

Byte Offset	Bits	Name	Description
0h	63:0	TDI_ID	Firmware provided identifier used by the guest to identify the TDI in guest messages.
8h	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.
Ch	31:1	–	Reserved.
	0	WRITE	0: Only retrieve configuration of range. 1: Write configuration of range.

When WRITE is 0, 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 83. Layout of the TIO_MSG_MMIO_CONFIG_RSP Structure

Byte Offset	Bits	Name	Description
0h	63:0	TDI_ID	Firmware provided identifier used by the guest to identify the TDI in guest messages.
8h	31:16	–	Reserved.
	15:0	STATUS	0: Success. 1: Invalid TDI 2: TDI is not bound to the guest. 3: The provided MMIO range ID is not reported in the interface report. 4: One or more attributes could not be changed. *All other encodings reserved.*
Ch	31:16	RANGE_ID	Identifier of the range.
	15:4	–	Reserved.
	3	IS_MEM_ATTR_UPDATEABLE	Indicates certain TDISP flags can be updated.
	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.

Byte Offset	Bits	Name	Description
	1	MSIX_PBA	Indicates if this range maps MSI-X PBA.
	0	MSIX_TABLE	Indicates if the range maps MSI-X table.
10h	31:1	–	Reserved.
	0	WRITE	0: Only retrieve configuration of range. 1: Write configuration of range.
14h	95:0	Reserved.	Must be zero

8.6 SDTE Write

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

Table 84. Layout of the TIO_MSG_SDTE_WRITE_REQ Structure

Byte Offset	Bits	Name	Description
0h	63:0	TDI_ID	Firmware provided identifier used by the guest to identify the TDI in guest messages.
8h-Fh		–	Reserved.
10h	511:0	SDTE	sDTE to use to configure the guest-controlled fields. See *Table* 85below.

Table 85. Layout of the SDTE Structure (bits not specified are reserved)

Bits	Name	Description
511:350	–	Reserved.
351:321	VIRTUAL_TOM	vTOM applied to all TDI accesses. This must point to a 2 MB page.
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.
207	vIMUEn	Set to 1 by firmware if VIOMMU_EN = 1 in SNP_INIT_EX options.
183:182	GPM	Guest paging mode options. See Table 7 of [IOMMU]
127:107	GCR3 Table Root Pointer	GPA of guest nested page table.
96	I	IOTLB enable. See Table 7 of [IOMMU]
62	IW	I/O write permission bit.
61	IR	I/O read permission bit.
60:58	GCR Table Root Pointer	Bits [14:12] of GCR3 Table Root pointer. See Table 7 of [IOMMU].
57:56	GLX	Guest Levels Translated. See Table 7 of [IOMMU]
55	GV	Guest Paging Mode Enable Bit. See Table 7 of [IOMMU]
54	GioV	Guest I/O protection valid. See Table 7 of [IOMMU]
52	PPR	Enable PPR Interface feature. See Table 7 of [IOMMU]

Bits	Name	Description
51:1	–	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 86. Layout of the TIO_MSG_SDTE_WRITE_RSP Structure

Byte Offset	Bits	Name	Description
0h	63:0	TDI_ID	Firmware provided identifier used by the guest to identify the TDI in guest messages.
8h	15:0	STATUS	0: Success. 1: Invalid TDI 2. TDI is unbound. 3: Reserved fields were not 0. 4: Operation Failure. *All other encodings reserved.*
Ah-Fh		–	Reserved.

Chapter 9 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.