Intel® Trust Domain Extensions (Intel® TDX) Module Architecture Application Binary Interface (ABI) Reference Specification

348551-005US

October 2024

Notices and Disclaimers

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

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

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

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

No product or component can be absolutely secure.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted that includes the subject matter disclosed herein.

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

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

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

Intel TDX Application Binary Interface (ABI) Reference

Table of Contents

Intel TDX Application Binary Interface (ABI) Reference

1. About this Document

1.1. Scope of this Document

This document describes the Application Binary Interface (ABI) of the Intel® Trust Domain Extensions (Intel® TDX) module, implemented using the Intel TDX Instruction Set Architecture (ISA) extensions, for confidential execution of Trust Domains in an untrusted hosted cloud environment.

This document is part of the **TDX Module Architecture Specification Set**, which includes the following documents:

Table 1.1: TDX Module Architecture Specification Set

Document Name	Reference	Description
TDX Module Base Architecture Specification	[TDX Module Base Spec]	Base TDX module architecture overview and specification, covering key management, TD lifecycle management, memory management, virtualization, measurement and attestation, service TDs, debug aspects etc.
TDX Module TD Migration Architecture Specification	[TD Migration Spec]	Architecture overview and specification for TD migration
TDX Module TD Partitioning Architecture Specification	[TD Partitioning Spec]	Architecture overview and specification for TD Partitioning
TDX Module TDX Connect Specification	[TDX Connect Spec]	Architecture overview and specification for TDX Connect
→ **TDX Module ABI Reference Specification**	[TDX Module ABI Spec]	Detailed TDX module Application Binary Interface (ABI) reference specification, covering the entire TDX module architecture
TDX Module TDX Connect ABI Reference Specification	[TDX Connect ABI Spec]	Detailed TDX module Application Binary Interface (ABI) reference specification, covering the TDX connect architecture
TDX Module ABI Reference Tables	[TDX Module ABI Tables]	A set of JSON format files detailing TDX module Application Binary Interface (ABI)
TDX Module ABI Incompatibilities	[TDX Module ABI Incompatibilities]	Description of the incompatibilities between TDX 1.0 and TDX 1.4/1.5 that may impact the host VMM and/or guest TDs

This document is a work in progress and is subject to change based on customer feedback and internal analysis. This document does not imply any product commitment from Intel to anything in terms of features and/or behaviors.

Note: The contents of this document are accurate to the best of Intel's knowledge as of the date of publication, though Intel does not represent that such information will remain as described indefinitely in light of future research and design implementations. Intel does not commit to update this document in real time when such changes occur.

1.2. Glossary

See the [TDX Module Base Spec].

1.3. Notation

See the [TDX Module Base Spec].

1.4. References

See the [TDX Module Base Spec].

Intel TDX Application Binary Interface (ABI) Reference

2. CPU Virtualization Tables

2.1. MSR Virtualization

Most of the MSR virtualization information is provided in a separate JSON format file **msr_virtualization.json**. Additional information about specific MSRs is provided below.

2.1.1. IA32_ARCH_CAPABILITIES (MSR 0x10A)

The virtualization of IA32_ARCH_CAPABILITIES (MSR 0x10A) is described in the [Base Spec] section "Checking and Virtualization of CPU Side Channel Protection Mechanisms Enumeration".

2.1.2. IA32_MISC_ENABLE (MSR 0x1A0)

The virtualization of IA32_MISC_ENABLE (MSR 0x1A0) depends on TDCS.TD_CTLS.REDUCE_VE, as set by the guest TD. Support of this bit is enumerated by TDX_FEATURES0.VE_REDUCTION (bit 30).

If TDCS.TD_CTLS.REDUCE_VE is 0 or is not supported, then RDMSR(IA32_MISC_ENABLE) returns the MSR's native value, except that if the TD's ATTRIBUTES.PERFMON is 0, then bit 7 (Perfmon Available) is set to 0 and bit 12 (PEBS Unavailable) is set to 1. WRMSR(IA32_MISC_ENABLE) results in a #VE(CONFIG_PARAVIRT).

If TDCS.TD_CTLS.REDUCE_VE is 1, then RDMSR(IA32_MISC_ENABLE) reads from a shadow value in TDVPS, and WRMSR(IA32_MISC_ENABLE) behaves as described in the table below.

Table 2.1: IA32_MISC_ENABLE (MSR 0x1A0) Virtualization when TDCS.TD_CTLS.REDUCE_VE is Set to 1

Bit	Name	Access	Init Value of Shadow (in TDVPS)	On RDMSR	On WRMSR	Description
0	Fast-Strings Enable	RW	Native value	From shadow	To shadow	
3	Automatic Thermal Control Circuit Enable	RW	0	From shadow	To shadow	
7	Perfmon Available	RO	ATTRIBUTES.PERFMON	From shadow	Ignore	
11	BTS Unavailable	RO	Native (checked to be 1)	From shadow	Ignore	
12	PEBS Unavailable	RO	Native & ~ATTRIBUTES.PERFMON	From shadow	Ignore	
16	Enhanced Speed Step	RW	Virt. CPUID(1).ECX[7]	From shadow	If (virt. CPUID(1).ECX[7] == 0) and (value == 1), #GP. Else, write to shadow	Support paravirtualization
18	Enable MONITOR FSM	RW	Virt. CPUID(1).ECX[3]	From shadow	If (value is being modified), #VE(UNSUPPORTED_FEATURE). Else, write to shadow	Guest TD is not expected to change this bit.
22	Limit CPUID Max Leaf	RW	0	From shadow	If (value == 1), #VE(UNSUPPORTED_FEATURE). Value in shadow remains 0.	Simplify CPUID handling, not supposed to happen with modern OS
23	xTPR Message Disable	RW	0	From shadow	If (virt. CPUID(1).ECX[14] == 0) and (value == 1), #GP. Else, write to shadow	TDs are not allowed to broadcast IPIs. Host VMM can control this bit.

Bit	Name	Access	Init Value of Shadow (in TDVPS)	On RDMSR	On WRMSR	Description
34	XD Bit Disable	RW	0	From shadow	If (value == 1), #GP. Value in shadow remains 0.	This bit is deprecated.
Other	Reserved	RO	0	From shadow	If (value == 1), #GP. Value in shadow remains 0.	

2.1.3. IA32_DEBUGCTL (MSR 0x1D9)

See the [Base Spec] section “On-TD Debug”.

2.1.4. IA32_X2APIC_* (MSRs 0x800 – 0x8FF)

See the [Base Spec] section “Virtual APIC Access by Guest TD”.

2.2. CPUID Virtualization

CPUID virtualization information is provided in a separate JSON format file **cpuid_virtualization.json**.

3. Data Types

This section describes data types that are designed to be used by the Intel TDX module.

3.1. Interface Function Completion Status

Note: This section provides a high-level overview of function completion status, as defined. Implementation details may differ.

A high-level definition of the interface functions completion status is provided in the [TDX Module Base Spec].

3.1.1. Function Completion Status Structure

Table 3.1: Intel TDX Interface Functions Completion Status (Returned in RAX) Definition

Bits	Name	Description
63	ERROR	Interface function aborted due to error. 0: Indicates that the function completed successfully – possibly with some warnings. 1: Indicates that the function aborted due to some error.
62	NON_RECOVERABLE	Recoverability hint – applicable only when ERROR is 1. 0: Indicates that the function may possibly be retried after some conditions have been corrected. 1: Indicates that the error is probably not recoverable.
61	FATAL	Fatality hint – applicable only for SEAMCALL. 0: Indicates that the TD can continue its normal lifecycle. 1: Indicates that the TD entered a state where it can only be torn down. E.g., when an import has failed and the TD's OP_STATE is FAILED_IMPORT.
60	HOST_RECOVERABILITY_HINT	As a TDH.VP.ENTER output, indicates a TDCALL that resulted in a trap-like TD exit for which the host VMM needs to provide a recoverability hint in the following TD entry. On the following TDH.VP.ENTER, the host VMM provides a hint to the guest TD, which is the output of the TDCALL: 0: The host VMM hints that the guest-side function may possibly be retried (e.g., the host may have corrected some conditions). 1: The host VMM hints that the error is probably not recoverable.
59:48	RESERVED	Reserved – set to 0
47:40	CLASS	Class of the function completion status
39:32	DETAILS_L1	Details of the function completion status
31:0	DETAILS_L2	Additional details of the function completion status – e.g., includes: • Implicit or explicit operand identifier • CPUID leaf or sub-leaf • MSR index • VMCS field code • VM exit reason • CMR index • TDMR index

3.1.2. Function Completion Status Code Classes (Bits 47:40)

Table 3.2: Function Completion Status Code Classes (Bits 47:40) Definition

Class ID	Class Name	Description
0	General	General function completion status
1	Invalid Operand	An invalid operand value has been provided, e.g., HKID is out of range, HPA overlaps SEAMRR, GPA is not private, etc.
2	Resource Busy	Resource is busy, there is a concurrency conflict.
3	Page Metadata	Page metadata (in PAMT) are incorrect, e.g., page type is wrong.
4	Dependent Resources	The state of dependent resources is incorrect, e.g., there are TD pages while trying to reclaim a TDR page.
5	Intel TDX Module State	The Intel TDX module state is incorrect.
6	TD State	The state of the TD is incorrect, e.g., it has not been initialized yet.
7	TD VCPU State	The state of the TD VCPU is incorrect, e.g., it is corrupted.
8	Key Management	The status code is related to key management, e.g., keys are not configured.
9	Platform	The status code is related to platform configuration or state.
10	Physical Memory	The status code is related to physical memory.
11	Guest TD Memory	The status code is related to guest TD memory.
12	Metadata	The status code is related to metadata (global scope, TD scope or VCPU scope)
13	Service TD	The status code is related to a service TD
14	Migration	The status code is related to TD migration
15	TDX I/O	The status code is related to TDX I/O
16	Measurement	The status code is related to TDX measurement
17	TD Partitioning	The status code is related to TD partitioning
255	Reserved	Reserved for use by host VMM or guest TD software This value is never used by the TDX module.

3.1.3. Function Completion Status Codes and Operand IDs

Interface functions completion status codes and operand IDs are provided in a separate JSON format file **interface_functions_completion_status.json**.

3.2. Basic Crypto Types

Table 3.3: Basic Crypto Types

Name	Size (Bytes)	Description
SHA384_HASH	48	384-bit buffer containing the result of a SHA384 hash calculation
KEY128	16	128-bit key

Name	Size (Bytes)	Description
KEY256	32	256-bit key

3.3. TDX Module Configuration, Enumeration and Initialization Types

Note: This section describes configuration, enumeration and initialization types, as defined. Implementation may differ.

3.3.1. CPUID_CONFIG

CPUID_CONFIG is designed to enumerate how the host VMM may configure the virtualization done by the Intel TDX module for a single CPUID leaf and sub-leaf. An array of CPUID_CONFIG entries is used for the Intel TDX module enumeration by TDH.SYS.INFO.

Table 3.4: CPUID_CONFIG Definition

Field	Offset (Bytes)	Size (Bytes)	Description
LEAF	0	4	EAX input value to CPUID
SUB_LEAF	4	4	ECX input value to CPUID A value of -1 indicates a CPUID leaf with no sub-leaves.
EAX	8	4	Enumeration of the configurable virtualization of the value returned by CPUID in EAX: a value of 1 in any of the bits indicates that the host VMM is allowed to configure that bit
EBX	12	4	Enumeration of the configurable virtualization of the value returned by CPUID in EBX: a value of 1 in any of the bits indicates that the host VMM is allowed to configure that bit
ECX	16	4	Enumeration of the configurable virtualization of the value returned by CPUID in ECX: a value of 1 in any of the bits indicates that the host VMM is allowed to configure that bit
EDX	20	4	Enumeration of the configurable virtualization of the value returned by CPUID in EDX: a value of 1 in any of the bits indicates that the host VMM is allowed to configure that bit

3.3.2. TDX Module Version

The TDX module version is enumerated as five fields, as shown in the table below. When written as a string, the fields are separated by a dot, e.g., “1.5.08.04.0234”.

Table 3.5: TDX Module Version Definition

Field	As Text	Size (Bytes)	Description
MAJOR_VERSION and MINOR_VERSION	1 digit each, e.g., “**1.5**”	16 bits each	Together, represent the main version number of the TDX module. Usually related to the supported SOCs.

Field	As Text	Size (Bytes)	Description
UPDATE_VERSION	2 digits, e.g., "1.5.**08**"	16 bits	A sub-version of the major/minor version. The update version number is incremented after every production-signed drop of the TDX module. E.g., if version 1.5.01 is production-signed, the next drop will be not use 01 as its update version, regardless of being a production-signed drop or not.
INTERNAL_VERSION	2 digits, e.g., "1.5.08.**04**"	16 bits	A sub-version of the update version. Denotes internal release number.
BUILD_NUM	4 digits, e.g., "1.5.08.04.**0234**"	16 bits	A unique build number

3.3.3. Global-Scope (TDX Module) Metadata

TDX module global scope fields provide enumeration information about the Intel TDX module. They are used with the TDH.SYS.RD, TDH.SYS.RDALL, TDG.SYS.RD and TDG.SYS.RDALL functions.

3.3.3.1. TDX Features Enumeration

The main enumeration of features supported by the TDX module is provided by the TDX_FEATURES array of 64-bit metadata fields. The number of fields is enumerated by NUM_TDX_FEATURES.

The TDX module features of TDX 1.0 are considered a baseline. TDX_FEATURES enumerate features beyond that baseline.

Table 3.6: TDX_FEATURES0 Definition

Bit(s)	Name	Description
0	TD_MIGRATION	The TDX module supports TD migration. Further information is provided by the Migration fields.
1	TD_PRESERVING	The TDX module supports TD preserving updates. Further information is provided by the TDX Module Handoff metadata fields.
2	SERVICE_TD	The TDX module supports Service TDs. Further information is provided by the Service TD fields.
3	ENHANCED_METADATA	The TDX module supports enhanced metadata interface functions: • Version 1 of previously existing functions: TDH.MNG.RD, TDH.VP.RD and TDG.VM.RD. • New functions: TDH.SYS.RD, TDH.SYS.RDALL, TDG.SYS.RD, TDG.SYS.RDALL, TDG.VP.RD/WR.

4	RELAXED_MEM_MNG	The TDX module's memory management requirements are relaxed vs. TDX 1.0:
		• Many interface functions allow concurrent memory management operations by exclusively locking specific Secure EPT entries instead of the whole Secure EPT tree:
		  ○ TDH.MEM.PAGE.AUG/DEMOTE/PROMOTE/RELOCATE/REMOVE
		  ○ TDH.MEM.SEPT.ADD
		• TLB tracking (e.g., TDH.MEM.RANGE.BLOCK followed by TDH.MEM.TRACK and IPIs) may be skipped if the TD's OP_STATE is such that the TD can't be running, i.e., in the following cases:
		  ○ On normal TD build, the TD's measurement has not yet been finalized by TDH.MR.FINALIZED.
		  ○ The TD has been paused for export by TDH.EXPORT.PAUSE, and export has not been aborted by TDH.EXPOR.ABORT.

Bit(s)	Name	Description
		○ On import, TD execution has been enabled by TDH.IMPORT.COMMIT or TDH.IMPORT.END.
5	CPUID_VIRT_GUEST_CTRL	Guest TD may request that on certain CPUID leaves/sub-leaves a #VE(CONFIG_PARAVIRT) will always be injected, using the TDG.VP.RD/WR to access the TDVPS' CPUID_CONTROL fields.
6	TDX_CONNECT	Both the TDX module and the CPU support TDX Connect.
7	TD_PARTITIONING	The TDX module supports TD partitioning: • New interface functions: TDG.MEM,PAGE.ATTR.RD/WR, TDG.VP.ENTER, TDG.VP.INVEPT, TDG.VP.INVGLA • Version 1 of existing interface functions: TDH.MEM.PAGE.PROMOTE, TDH.MEM.SEPT.ADD, TDH.MEM.SEPT.REMOVE • Backward-compatible updates to existing interface functions, with new input and/or output operands.
8	LOCAL_ATTESTATION	The TDX module supports local attestation: • New interface function: TDG.MR.VERIFYREPORT
9	TD_ENTRY_ENHANCEMENTS	The TDX module supports the following TD entry enhancements: • HOST_RECOVERABILITY_HINT: On trap-like asynchronous TD exit, bit 60 of the TDH.VP.ENTER completion status (returned in RAX) may be set to 1. In this case, the host VMM may set the following TD entry's input value of RCX' HOST_RECOVERABILITY_HINT bit; this bit is copied to the guest RAX bit 60, which the guest interprets as part of a TDCALL completion status.
10	HOST_PRIORITY_LOCKS	The TDX module implements host-priority locks to avoid denial-of-service by guest TDs. This requires the host VMM to retry operations that fail with a TDX_OPERAND_BUSY status.
11	CONFIG_IA32_ARCH_CAPABILITIES	The TDX module allows the host VMM to configure the virtualization of IA32_ARCH_CAPABILITIES MSR.
12	SEALING	The TDX module supports signed TDs and seal keys bound to new TD properties: • New interface functions: TDG.MR.KEY.GET • SEALING support depends on TD_SIGNING_AND_SVN (bit 22) support
13	S4	The TDX module supports state hibernation and restoration across S4 CPU state: • New interface function: TDH.SYS.S4_END • New input flag to TDH.EXPORT/IMPORT.STATE.IMMUTABLE
14	ACT	The TDX module manages memory access control using the CPU's Access Control Table (ACT).
15	WBINVD_DOMAINS	TDH.PHYMEM.CACHE.WB needs to be called per WBINVD domain that might be different than a whole package. WBINVD domains are enumerated with the WBINVD_DOMAIN* metadata fields. If SKIP_PHYMEM_CACHE_WB (bit 34) is 1, then WBINVD_DOMAINS is 0.
16	PENDING_EPT_VIOLATION_V2	The TDX module supports enhanced handling of EPT violation on guest TD access to PENDING pages: • Decision on whether a #VE(PENDING) is injected to the guest TD can be guest configurable. • Extended exit qualification is provided to the host VMM. • EPT violation on L2 VM access to a PENDING page always causes an L2→L1 exit.

Bit(s)	Name	Description
17	FMS_CONFIG	The TDX module supports configuration of virtual CPUID(1).EAX (Family/Model/Stepping) value for migratable TDs.
18	NO_RBP_MOD	The TDX module supports a TD configuration where RBP is never modified by any host-side (SEAMCALL) and guest-side (TDCALL) interface function. This is configured by TD_PARAMS.CONFIG_FLAGS.NO_RBP_MOD (see 3.4.3).
19	L2_TLB_INVD_OPT	The TDX module supports additional address translation invalidation modes on TDG.VP.ENTER.
20	TOPOLOGY_ENUM	The TDX module supports virtual topology enumeration and configuration of CPUID(0xB), CPUID(0x1F) and x2APIC ID.
21	PARTITIONED_TD_MIGRATION	The TDX module supports migration of partitioned TDs.
22	TD_SIGNING_AND_SVN	The TDX module supports the following: • REPORTTYPE.VERSION value of 2 • TDSIGSTRUCT • Additional attestation configuration (MRSIGNER, PRODID, etc.) • TDG.MR.ASSIGNSVNS interface function
23	CLFLUSH_BEFORE_ALLOC	When allocating a memory page to be used as TD private memory or TD control structure page, the host VMM is required to ensure that none of the cache lines associated with the page is in a MODIFIED state.
24	EVENT_FILTERING	The TDX module supports filtering of performance monitoring events, based on configuration by the host VMM as part of TDH.MNG.INIT.
25	ICSSD	Instruction-Count based Single-Step Defense: Indicates that the TDX module supports single-step attack detection based on counting TD VCPU instructions. This feature is only available for guest TDs where performance monitoring is not enabled (ATTRIBUTES.PERFMON == 0).
26	FIXED_CTR12_PROF	System profiling by IA32_FIXED_CTR1 and IA32_FIXED_CTR2 is supported. IA32_FIXED_CTR1 and IA32_FIXED_CTR2 continue counting while the TDX module is running. If a TD is not enabled for performance monitoring (ATTRIBUTES.PERFMON == 0) and not debuggable (ATTRIBUTES.DEBUG == 0) then the counters continue counting while that TD is running.
27	MAXPA_VIRT	The TDX module supports virtualization of physical address width, as enumerated by CPUID(0x80000008).EAX[7:0].
28	APX	Both the TDX module and the CPU support Intel® APX (Advanced Performance Extensions).
29	CPUID2_VIRT	The TDX module supports virtualization of CPUID(2)
30	VE_REDUCTION	The TDX module supports run time controls by the guest TD to reduce the cases where #VE is injected by the TDX module on guest TD execution of CPUID, RDMSR/WRMSR and other instructions. **Note:** VE_REDUCTION implies TOPOLOGY_ENUM and CPUID2_VIRT.
31	ENHANCED_EVENT_FILTERING	The TDX module supports enhanced filtering of performance monitoring events, based on configuration by the host VMM as part of TDH.MNG.INIT.
32	TDX_CONNECT_PARTITIONING	The TDX module supports TDX Connect for partitioned TDs.
33	MAXGPA_VIRT	The TDX module supports virtualization of guest physical address width, as enumerated by CPUID(0x80000008).EAX[23:16].

Bit(s)	Name	Description
34	SKIP_PHYMEM_CACHE_WB	The host VMM needs not call TDH.PHYMEM.CACHE.WB as part of the TD teardown sequence.
35	NON_BLOCKING_RESIZE	The TDX module supports TDH.MEM.PAGE.DEMOTE and TDH.MEM.PAGE.PROMOTE without blocking and TLB tracking.
63:36	RESERVED	Set to 0

3.3.3.2. Global Metadata Fields

Global metadata information is provided in a separate JSON format file **global_metadata.json**.

3.3.4. CMR_INFO

CMR_INFO is designed to provide information about a Convertible Memory Range (CMR), as configured by BIOS and checked and stored securely by MCHECK.

Note: CMR_INFO and TDH.SYS.INFO are provided for backward compatibility. TDH.SYS.RDALL is the recommended method to read Intel TDX module information. See also 3.3.3 above.

Table 3.7: CMR_INFO Entry Definition

Name	Offset (Bytes)	Type	Size (Bytes)	Description
CMR_BASE	0	Physical Address	8	Base address of the CMR: since a CMR is aligned on 4KB, bits 11:0 are 0.
CMR_SIZE	8	Integer	8	Size of the CMR, in bytes: since a CMR is aligned on 4KB, bits 11:0 are 0. A value of 0 indicates a null entry.

TDH.SYS.INFO leaf function returns an array of CMR_INFO entries. The CMRs are sorted from the lowest base address to the highest base address, and they are non-overlapping.

3.3.5. TDSYSINFO_STRUCT

TDSYSINFO_STRUCT is designed to provide enumeration information about the Intel TDX module. It is an output of the TDH.SYS.INFO leaf function.

Note: TDSYSINFO_STRUCT and TDH.SYS.INFO are provided for backward compatibility. TDH.SYS.RDALL is the recommended method to read Intel TDX module information. See also 3.3.3 above.

TDSYSINFO_STRUCT's size is 1024B.

Table 3.8: TDSYSINFO_STRUCT Definition

Section	Field Name	Offset (Bytes)	Type	Size (Bytes)	Description
Intel TDX Module Info	**ATTRIBUTES**	0	Bitmap	4	Module attributes **Bits 30:0** Reserved – set to 0 **Bit 31** 0 indicates a production module. 1 indicates a debug module.
	VENDOR_ID	4	Integer	4	0x8086 for Intel

Section	Field Name	Offset (Bytes)	Type	Size (Bytes)	Description
	BUILD_DATE	8	BCD	4	Intel TDX module build data – in yyyymmdd BCD format (each digit occupies 4 bits)
	BUILD_NUM	12	Integer	2	Build number of the Intel TDX module
	MINOR_VERSION	14	Integer	2	Minor version number of the Intel TDX module
	MAJOR_VERSION	16	Integer	2	Major version number of the Intel TDX module
	SYS_RD	18	Boolean	1	A non-0 value indicates that the information in this structure is incomplete. TDH.SYS.RD or TDH.SYS.RDALL should be used to obtain TDX module information.
	RESERVED	19	N/A	13	This field is reserved for enumerating future Intel TDX module capabilities. Set to 0
Memory Info	**MAX_TDMRS**	32	Integer	2	The maximum number of TDMRs supported
	MAX_RESERVED_ PER_TDMR	34	Integer	2	The maximum number of reserved areas per TDMR
	PAMT_ENTRY_ SIZE	36	Integer	2	The size of a PAMT entry – determines the number of bytes that need to be reserved for the three PAMT areas: • PAMT_1G (1 entry per 1GB of TDMR) • PAMT_2M (1 entry per 2MB of TDMR) • PAMT_4K (1 entry per 4KB of TDMR)
	RESERVED	38	N/A	10	Set to 0
Control Struct Info	**TDCS_BASE_SIZE**	48	Integer	2	Base value for the number of bytes required to hold TDCS
	RESERVED	50	N/A	2	Reserved for additional TDCS enumeration Set to 0
	TDVPS_BASE_SIZE	52	Integer	2	Base value for the number of bytes required to hold TDVPS

Section	Field Name	Offset (Bytes)	Type	Size (Bytes)	Description
	RESERVED	54	N/A	10	Set to 0
TD Capabilities	**ATTRIBUTES_ FIXED0**	64	Bitmap	8	If any certain bit is 0 in ATTRIBUTES_FIXED0, it must be 0 in any TD's ATTRIBUTES. The value of this field reflects the Intel TDX module capabilities and configuration and CPU capabilities.
	ATTRIBUTES_ FIXED1	72	Bitmap	8	If any certain bit is 1 in ATTRIBUTES_FIXED1, it must be 1 in any TD's ATTRIBUTES. The value of this field reflects the Intel TDX module capabilities and configuration and CPU capabilities.
	XFAM_FIXED0	80	Bitmap	8	If any certain bit is 0 in XFAM_FIXED0, it must be 0 in any TD's XFAM.
	XFAM_FIXED1	88	Bitmap	8	If any certain bit is 1 in XFAM_FIXED1, it must be 1 in any TD's XFAM.
	RESERVED	96	N/A	32	Set to 0
	NUM_CPUID_ CONFIG	128	Integer	4	Number of the following CPUID_CONFIG entries
	CPUID_CONFIG[0]	132	CPUID_ CONFIG	24	Enumeration of the CPUID leaves/sub-leaves that contain bit fields whose virtualization by the Intel TDX module is either: • Directly configurable (CONFIG_DIRECT) by the host VMM • Bits that the host VMM may allow to be 1 (ALLOW_*_DIRECT) and their native value, as returned by the CPU, is 1. See 3.3.1 for details. Note that the virtualization of many CPUID bit fields not enumerated in this list is configurable indirectly via the XFAM and ATTRIBUTES assigned to a TD by the host VMM.
	CPUID_ CONFIG[last]		CPUID_ CONFIG	24	
Reserved	**RESERVED**		N/A		Fills up to the structure size (1024B) – set to 0

3.3.6. TDMR_INFO

TDMR_INFO is designed to provide information about a single Trust Domain Memory Region (TDMR) and its associated PAMT. It is used as an input to TDH.SYS.CONFIG.

Table 3.9: TDMR_INFO Entry Definition

Name	Offset (Bytes)	Type	Size (Bytes)	Description
TDMR_BASE	0	Physical Address	8	Base address of the TDMR (HKID bits must be 0): since a TDMR is aligned on 1GB, bits 29:0 are always 0.
TDMR_SIZE	8	Integer	8	Size of the TDMR, in bytes: must be greater than 0 and a whole multiple of 1GB (i.e., bits 29:0 are always 0).
PAMT_1G_BASE	16	Physical Address	8	Base address of the PAMT_1G range associated with the above TDMR (HKID bits must be 0): since a PAMT range is aligned on 4KB, bits 11:0 are always 0.
PAMT_1G_SIZE	24	Integer	8	Size of the PAMT_1G range associated with the above TDMR: since a PAMT range is aligned on 4KB, bits 11:0 are always 0.
PAMT_2M_BASE	32	Physical Address	8	Base address of the PAMT_2M range associated with the above TDMR (HKID bits must be 0): since a PAMT range is aligned on 4KB, bits 11:0 are always 0.
PAMT_2M_SIZE	40	Integer	8	Size of the PAMT_2M range associated with the above TDMR: since a PAMT range is aligned on 4KB, bits 11:0 are always 0.
PAMT_4K_BASE	48	Physical Address	8	Base address of the PAMT_4K range associated with the above TDMR (HKID bits must be 0): since a PAMT range is aligned on 4KB, bits 11:0 are always 0.
PAMT_4K_SIZE	56	Integer	8	Size of the PAMT_4K range associated with the above TDMR: since a PAMT range is aligned on 4KB, bits 11:0 are always 0.
RESERVED_OFFSET[0]	64	Integer	8	• Offset of reserved range 0 within the TDMR: since a reserved range is aligned on 4KB, bits 11:0 are always 0.
RESERVED_SIZE[0]	72	Integer	8	Size of reserved range 0 within the TDMR: • A size of 0 indicates a null entry. All following reserved range entries must also be null. • Since a reserved range is aligned on 4KB, bits 11:0 are always 0.
RESERVED_OFFSET[N-1]	64 + 16*(N-1)	Integer	8	Offset of the last reserved range within the TDMR.
RESERVED_SIZE[N-1]	72 + 16*(N-1)	Integer	8	Size of the last reserved range within the TDMR.

5

Notes:

- The number of reserved areas within a TDMR is enumerated by TDX Module's MAX_RESREVED_PER_TDMR metadata field, which can be read using TDH.SYS.RD, TDH.SYS.RDALL or TDH.SYS.RDM. For details, see 3.3.53.3.3.

- For backward compatibility, this value is also enumerated by TDSYSINFO_STRUCT.MAX_RESREVED_PER_TDMR (see 3.3.5).
- Within each TDMR entry, all reserved areas must be sorted from the lowest offset to the highest offset, and they must not overlap with each other.
- All TDMRs and PAMTs must be contained within CMRs.
- A PAMT area must not overlap with another PAMT area (associated with any TDMR), and it must not overlap with non-reserved areas of any TDMR. PAMT areas may reside within reserved areas of TDMRs.

3.4. TD Parameter Types

Note: This section describes TD parameter types, as defined. Implementation details may differ.

3.4.1. ATTRIBUTES

ATTRIBUTES is defined as a 64b field that specifies various attested guest TD attributes. ATTRIBUTES is provided by the host VMM as a guest TD initialization parameter as part of TD_PARAMS. It is reported to the guest TD by TDG.VP.INFO, TDG.VM.RD* and as part of TDREPORT_STRUCT returned by TDG.MR.REPORT. ATTRIBUTES is migrated to a destination platform as part of the immutable TD state export by TDH.EXPORT.STATE.IMMUTABLE and import by TDH.IMPORT.STATE.IMMUTABLE.

The ATTRIBUTES bits are divided into four groups, as shown in the table below, according to their impact on TD security:

- If any bit in the TUD group is set to 1, the guest TD is under off-TD debug and is untrusted.
- Bits in the TUP group indicate features that impact security and trust. It is up to the remote verifier to decide whether the impact on TD trustworthiness is acceptable.
- Bits in the SEC group indicate features that may impact TD security but are not considered as impacting TD trust. Bits in the SEC group may have a positive or a negative impact on the TD security if set, as specified in the table.
- Bits in the OTHER group indicate feature that are attested but do not impact TD security.

The table below shows the whole set of ATTRIBUTES bits that have been defined. However, the following must be noted:

- Some versions of the TDX module may not support some of the ATTRIBUTES bits. E.g., for a TDX Module that does not support TD Migration, the MIGRATABLE bit must always be 0.
- Some of the ATTRIBUTES bits depend on CPU support. E.g., for a CPU does not support Key Locker, the KL bit must be 0.

Notes

- The host VMM can determine the supported set of ATTRIBUTES bits by reading the ATTRIBUTES_FIXED0 and ATTRIBUTES_FIXED1 fields using TDH.SYS.RD/RDALL.
- The attestation infrastructure may evaluate a TD's ATTRIBUTES based on the negative or positive security impact of each bit. For example, if a new version of the TDX module uses the currently reserved bit 25, the attestation infrastructure can know that this bit has a negative security impact when set, even without knowing the meaning of the bit.
- TD configuration that does not need to be attested (normally because it doesn't impact TD security) is not included in ATTRIBUTES. See the definition of CONFIG_FLAGS, CPUID configuration and other fields of TD_PARAMS in the following sections.

Table 3.10: ATTRIBUTES Definition

Bits	Group	Description	Bits	Bit Name	TD Security Impact if 1	Description
3:0	**TUD**	**TD Under Debug** If any of the bits in this group are set to 1, the guest TD is untrusted.	0	**DEBUG**	Negative	Guest TD runs in off-TD debug mode. Its VCPU state and private memory are accessible by the host VMM. DEBUG may not be set if MIGRATABLE is set.
			3:1	**RESERVED**	Negative	Reserved for future TUD flags – must be 0

Bits	Group	Description	Bits	Bit Name	TD Security Impact if 1	Description
15:4	**TUP**	**TD Under Profiling** The TD is subject to profiling, which may expose side channel information to untrusted entities.	4	**HGS_PLUS_PROF**	Negative	The TD is subject to HGS+ operation. HGS+ monitors the TD operation as part of the whole system. This bit may be set, if supported by the TDX module, regardless on CPU support.
			5	**PERF_PROF**	Negative	The TD is subject to system profiling using performance monitoring counters. Those counters are not context-switched on TD entry and exit; they monitor the TD operation as part of the whole system. This bit may be set, if supported by the TDX module, regardless on CPU support.
			6	**PMT_PROF**	Negative	The TD is subject to system profiling using core out-of-band telemetry. Core telemetry monitors the TD operation as part of the whole system. This bit may be set, if supported by the TDX module, regardless of CPU support.
			15:7	**RESERVED**	Negative	Reserved for future TUP flags – must be 0
31:16	**SEC**	**Security** Attributes that may impact TD security	16	**ICSSD**	Positive	Indicates that the TDX module must use Instruction-Count based Single-Step Defense to protect against single-step attacks. ICSSD may not be set if PERFMON is set. This bit may only be set if the TDX module supports ICSSD.
			22:17	**RESERVED_P**	Positive	Reserved for future SEC flags that will indicate positive impact on TD security. • As an input to TDH.MN.INIT, must be 0. Attestation verifiers may allow any value.
			26:23	**RESERVED_N**	Negative	Reserved for future SEC flags that will indicate negative impact on TD security – must be 0
			27	**LASS**	Positive	TD is allowed to use Linear Address Space Separation. This bit may only be set if both the TDX module and the CPU support LASS.
			28	**SEPT_VE_DISABLE**	Negative	Disable EPT violation conversion to #VE(PENDING) on guest TD access of PENDING pages
			29	**MIGRATABLE**	Negative	TD is migratable (using a Migration TD). MIGRATABLE may not be set if either DEBUG or PERFMON is set. This bit may only be set if the TDX module supports TD Migration.
			30	**PKS**	Positive	TD is allowed to use Supervisor Protection Keys. This bit may only be set if both the TDX module and the CPU support PKS.

Bits	Group	Description	Bits	Bit Name	TD Security Impact if 1	Description
			31	**KL**	Positive	TD is allowed to use Key Locker. This bit may only be set if both the TDX module and the CPU support Key Locker.
55:32	**RESERVED**	Reserved	55:32	**RESERVED**	None	Reserved for future expansion of the SEC group - must be 0
63:56	**OTHER**	Attributes that are attested but do not impact TD security	61:56	**RESERVED**	None	Reserved for future OTHER flags – must be 0
			62	**TPA**	None	The TD is a TDX Connect Provisioning Agent. This bit may only be set if both the TDX module and the CPU support TDX Connect.
			63	**PERFMON**	None	TD is allowed to use Perfmon and PERF_METRICS capabilities. PERFMON may not be set if either MIGRATABLE or ICSSD is set. This bit may only be set if the TDX Module supports Performance Monitoring virtualization.

3.4.2. XFAM

Intel SDM, Vol. 1, 13	Managing State Using the XSAVE Feature Set
Intel SDM, Vol. 3, 13	System Programming for Instruction Set Extensions and Processor Extended State

Intel TDX module extended state handling is described in the [TDX Module Base Spec].

XFAM (eXtended Features Available Mask) is defined as a 64b bitmap, which has the same format as XCR0 or IA32_XSS MSR. XFAM determines the set of extended features available for use by the guest TD. XFAM is provided by the host VMM as a guest TD initialization parameter as part of TD_PARAMS. It is reported to the guest TD by CPUID(0x0D, 0x01) and as part of TDREPORT_STRUCT returned by TDG.MR.REPORT.

The Intel TDX module and the Intel® Architecture impose some rules on how the bits of XFAM may be set. See the [TDX Module Base Spec] for details.

Support of XFAM bits depend on CPU support and TDX module support. The supported bit values can be enumerated by reading the XFAM_FIXED_0 and XFAM_FIXED_1 fields using TDH.SYS.RD/RDALL.

3.4.3. CONFIG_FLAGS

CONFIG_FLAGS is a set of TD configuration flags.

Table 3.11: TD_PARAMS_STRUCT.CONFIG_FLAGS Definition

Bits	Name	Description
0	**GPAW**	GPAW (Guest Physical Address Width[1]) controls the position of the SHARED bit in GPA. It is copied to each TD VMCS and L2 VMCS GPAW execution control on TDH.VP.INIT and TDH.IMPORT.STATE.VP. 0: GPA.SHARED bit is GPA[47] 1: GPA.SHARED bit is GPA[51] A value of 1 can only be specified if EPTP_CONTROLS[5:3] is specified as 4 (i.e., 5-level EPT). For details, see the [TDX Arch Extensions Spec].
1	**FLEXIBLE_PENDING_VE**	Controls the guest TD's ability to change the PENDING page access behavior from its default value: 0: The guest TD cannot change the behavior set by ATTRIBUTES.SEPT_VE_DISABLE. 1: The guest TD can change the default behavior set by ATTRIBUTES.SEPT_VE_DISABLE. **Enumeration:** Availability of FLEXIBLE_PENDING_VE is enumerated by TDX_FEATURES0.PENDING_EPT_VIOLATION_V2 (bit 16) and by CONFIG_PARAMS_FIXED0/1, readable using TDH.SYS.RD*.
2	**NO_RBP_MOD**	Controls whether RBP value can be modified by TDG.VP.VMCALL and TDH.VP.ENTER: 0: RBP can be used as an input to TDG.VP.VMCALL. The value provided by the guest TD is used as an output of TDH.VP.ENTER. The value provided by the host TD to the following TDH.VP.ENTER is used as an output of TDG.VP.VMCALL. 1: RBP can't be used as an input to TDG.VP.VMCALL. TDG.VP.VMCALL preserves the guest TD's value of RBP. TDH.VP.ENTER preserves the host VMM's value of RBP. **Enumeration:** Availability of NO_RBP_MOD is enumerated by TDX_FEATURES0.NO_RBP_MOD (bit 18) and by CONFIG_PARAMS_FIXED0/1, readable using TDH.SYS.RD*.
3	**MAXPA_VIRT**	Controls virtualization of physical address width, as enumerated by CPUID(0x80000008).EAX[7:0]: 0: The virtual value of CPUID(0x80000008).EAX[7:0] is set to the native value of that field. The virtual value of CPUID(0x80000008).EAX[23:16] is determined by the setting of MAXGPA_VIRT (bit 4 below). 1: MAXGPA_VIRT (bit 4 below) must be set to 0. The virtual value of CPUID(0x80000008).EAX[7:0] is configured by the host VMM. The virtual value of CPUID(0x80000008).EAX[23:16] is set to 0. For details, see the [Base Spec] discussion of GPA space size virtualization. **Enumeration:** Availability of MAXPA_VIRT is enumerated by TDX_FEATURES0.MAXPA_VIRT (bit 27) and by CONFIG_PARAMS_FIXED0/1, readable using TDH.SYS.RD*.

[1] The name is misleading, since the GPA width is not determined by GPAW.

Bits	Name	Description
4	**MAXGPA_VIRT**	Controls virtualization of guest physical address width, as enumerated by CPUID(0x80000008).EAX[23:16]: 0: The virtual value of CPUID(0x80000008).EAX[7:0] is determined by the setting of MAXPA_VIRT (bit 3 above). The virtual value of CPUID(0x80000008).EAX[23:16] is set to 0. 1: MAXPA_VIRT (bit 3 above) must be set to 0. The virtual value of CPUID(0x80000008).EAX[7:0] is set to the native value of that field. The virtual value of CPUID(0x80000008).EAX[23:16] is set depending on the value of GPAW (bit 0 above) and the native value of CPUID(0x80000008).EAX[7:0]. For details, see the [Base Spec] discussion of GPA space size virtualization. **Enumeration:** Availability of MAXGPA_VIRT is enumerated by TDX_FEATURES0.MAXGPA_VIRT (bit 33) and by CONFIG_PARAMS_FIXED0/1, readable using TDH.SYS.RD*.
63:5	**RESERVED**	Must be 0

3.4.4. CPUID_VALUES

CPUID_VALUES is defined as a 128b structure composed of four 32b fields representing the values returned by CPUID in registers EAX, EBX, ECX and EDX. An array of CPUID_RET is used during guest TD configuration by TDH.MNG.INIT.

Table 3.12: CPUID_VALUES Definition

Field	Offset (Bytes)	Size (Bytes)	Description
EAX	0	4	Value returned by CPUID in EAX
EBX	4	4	Value returned by CPUID in EBX
ECX	8	4	Value returned by CPUID in ECX
EDX	12	4	Value returned by CPUID in EDX

3.4.5. TD_PARAMS

TD_PARAMS is provided as an input to TDH.MNG.INIT, and some of its fields are included in the TD report. The format of this structure is valid for a specific MAJOR_VERSION of the Intel TDX module, as reported by TDH.SYS.RD/RDALL or TDH.SYS.INFO.

TD_PARAMS' size is 1024B.

Table 3.13: TD_PARAMS Definition

Field	Offset (Bytes)	Type	Size (Bytes)	Description	Included in TDREPORT?
ATTRIBUTES	0	64b bitmap (see 3.4.1)	8	TD attributes: the value set in this field must comply with ATTRIBUTES_FIXED0 and ATTRIBUTES_FIXED1 enumerated by TDH.SYS.RD/RDALL or TDH.SYS.INFO.	Yes

Field	Offset (Bytes)	Type	Size (Bytes)	Description	Included in TDREPORT?
XFAM	8	64b bitmap in XCR0 format	8	Extended Features Available Mask: indicates the extended state features allowed for the TD. XFAM's format is the same as XCR0 and IA32_XSS MSR. The value set in this field must satisfy the following conditions: • Natively valid value for XCR0 and IA32_XSS (does not contain reserved bits, features not supported by the CPU, or invalid bit combinations) • Complies with XFAM_FIXED0 and XFAM_FIXED1 as enumerated by TDH.SYS.RD/RDALL or TDH.SYS.INFO.	Yes
MAX_VCPUS	16	Unsigned 16b Integer	2	Maximum number of VCPUs Must be higher than 0. Must not be higher than MAX_VCPUS_PER_TD, which may be read by TDH.SYS.RD*.	No
NUM_L2_VMS	18	Unsigned 8b Integer	1	Number of L2 VMs May be between 0 and 3. A value of 0 indicates no TD Partitioning is supported.	No
MSR_CONFIG_CTLS	19	8b bitmap	1	MSR configuration controls: **Bit** — **Description** 0 — Indicates that TD configuration should use the IA32_ARCH_CAPABILITIES_CONFIG field below Other — Reserved, must be 0	No
RESERVED	20	N/A	4	Must be 0	No
EPTP_CONTROLS	24	EPTP	8	Control bits of EPTP – copied to each TD VMCS on TDH.VP.INIT: **Bits 2:0** Memory type – must be 110 (WB) **Bits 5:3** EPT level – 1 less than the EPT page-walk length. Must be either 3 or 4. Must comply with the EPT page-walk length supported by the CPU. **Bits 63:6** Reserved – must be 0	No
CONFIG_FLAGS	32	64b bitmap	8	Non-measured TD-scope execution controls. See 3.4.3 above for details.	No
TSC_FREQUENCY	40	16b unsigned integer	2	TD-scope virtual TSC frequency in units of 25MHz – must be between 4 and 400.	No
RESERVED	42	N/A	38	Must be 0	No
MRCONFIGID	80	SHA384_HASH	48	Software-defined ID for non-owner-defined configuration of the guest TD – e.g., run-time or OS configuration	Yes

Field	Offset (Bytes)	Type	Size (Bytes)	Description	Included in TDREPORT?
MROWNER	128	SHA384_HASH	48	Software-defined ID for the guest TD's owner	Yes
MROWNERCONFIG	176	SHA384_HASH	48	Software-defined ID for owner-defined configuration of the guest TD – e.g., specific to the workload rather than the run-time or OS	Yes
IA32_ARCH_ CAPABILITIES_CONFIG	224	64b bitmap	8	Configuration of IA32_ARCH_CAPABILITIES MSR virtualization (if enabled by MSR_CONFIG_CTLS above). Configuration capabilities are enumerated by the IA32_ARCH_CAPABILITIES_CONFIG_MASK. which can be read by TDH.SYS.RD/RDALL.	No
MRCONFIGSVN	232	16b unsigned integer	2	SVN corresponding to MRCONFIGID Support of this field is enumerated by TDX_FEATURES0.SEALING (bit 12). If not supported, this field must be 0.	Yes
MROWNERCONFIGSVN	234	16b unsigned integer	2	SVN corresponding to MROWNERCONFIG Support of this field is enumerated by TDX_FEATURES0.SEALING (bit 12). If not supported, this field must be 0.	Yes
RESERVED	236	N/A	20	Must be 0	No
CPUID_CONFIG[0]	256	CPUID_VALUES	16	Direct configuration of CPUID leaves/sub-leaves virtualization: the number and order of entries must be equal to the number and order of directly configurable or allowable CPUID leaves/sub-leaves reported by TDH.SYS.RD/RDALL or TDH.SYS.INFO. Note that the leaf and sub-leaf numbers are implicit. Only bits that have been reported as 1 by TDH.SYS.RD/RDALL or TDH.SYS.INFO may be set to 1. Note that the virtualization of many CPUID bit fields not enumerated in this list is configurable indirectly, via the XFAM and ATTRIBUTES fields.	No
...					
CPUID_CONFIG[n-1]		CPUID_VALUES	16		
RESERVED		N/A		Fills up to TD_PARAMS size (1024B) – must be 0	No

3.4.6. EVENT_FILTER and the EVENT_FILTERS Array

Enumeration: Support of EVENT_FILTER is enumerated by TDX_FEATURES0.EVENT_FILTERING (bit 24) and TDX_FEATURES0.ENHANCED_EVENT_FILTERING (bit 31), readable by TDH.SYS.RD*.

5 **EVENT_FILTER Entry**

EVENT_FILTER specifies a single criterion for filtering values written by the guest TD to the IA32_PERFEVTSELx MSRs.

Table 3.14: EVENT_FILTER Entry

Bits	Name	Description
7:0	EVENT_SELECT	Value for matching the IA32_PERFEVTSEL MSR's EVENT_SELECT field
30:8	RESERVED	Must be 0
31	NEGATIVE	If the TDX module supports enhanced event filtering, as enumerated by TDX_FEATURES0.ENHANCED_EVENT_FILTERING, then NEGATIVE indicates a negative match. Else NEGATIVE must be 0.
47:32	UMASK	Value for matching the IA32_PERFEVTSEL MSR's UMASK2 and UMASK field, after applying UMASK_MASK (if applicable). If the CPU does not support UMASK2, then the upper 8 bit of UMASK must be 0.
63:48	UMASK_MASK	If the TDX module supports enhanced event filtering, as enumerated by TDX_FEATURES0.ENHANCED_EVENT_FILTERING, then UMASK_MASK selects which bits of the IA32_PERFEVTSEL MSR's UMASK2 and UMASK fields to compare with UMASK. Else, UMASK_MASK must be 0xFFFF.

EVENT_FILTERS Array

EVENT_FILTERS is an array of EVENT_FILTER entries, provided by the host VMM as an input to TDH.MNG.INIT.

If the TDX module supports enhanced event filtering, as enumerated by TDX_FEATURES0.ENHANCED_EVENT_FILTERING, then the array must be sorted in an ascending order by EVENT_SELECT. Else, the array must be sorted in an ascending order by the raw 64-bit value of each entry and must not contain duplicate entries.

The maximum number of entries in the array is enumerated by MAX_EVENT_FILTERS, readable by TDH.SYS.RD*.

3.5. Physical Memory Management Types

Note: This section describes physical memory types, as defined. Implementation may differ.

PAMT entry and PT (page type) are defined in the [TDX Module Base Spec].

3.5.1. PAMT Page Type (PT) Values

Some PT values are applicable only when enumerated by certain TDX_FEATURES bits (see 3.3.3.1). If a certain PT value is not applicable, then it is considered reserved. For a detailed description of the page types, refer to the [Base Spec].

Table 3.15: PAMT Page Type Values

Page Type	Value	TDX_FEATURES Enumeration	
PT_NDA	0	N/A	The physical page is **Not Directly Assigned** to the Intel TDX module.
PT_RSVD	1	N/A	The physical page is reserved for non-TDX usage.
PT_PR	2	ACT (bit 14)	The physical page holds a page that is pending release.
PT_REG	3	N/A	The physical page holds TD private memory.
PT_TDR	4	N/A	The physical page holds the TD Root (TDR) control structure.
PT_TDCX	5	N/A	The physical page holds a TD control structure.
PT_TDVPR	6	N/A	The physical page holds a TD VCPU Root (TDVPR) page.

Page Type	Value	TDX_FEATURES Enumeration	
PT_TR	7	NON_BLOCKING_RESIZE (bit 35)	The physical page has no current GPA mapping, but the CPU may still hold TLB entries associated with it. The page must be TLB tracked before it can be assigned for any usage.
PT_EPT	8	N/A	The physical page holds a Secure EPT page.
PT_DEVIFCS_R	9	TDX_CONNECT (bit 6)	The physical page holds a DEVIFCS structure
PT_DEVIF_NR	10	TDX_CONNECT (bit 6)	The physical page holds a DEVIFCS internal data
PT_IOMMU_MT	11	TDX_CONNECT (bit 6)	The physical page holds an I/O (IOMMU/IDE/SPDM) data
PT_MMIO_MT	12	TDX_CONNECT (bit 6)	The physical page holds an MMIO metadata table
PT_DEVIFMT	13	TDX_CONNECT (bit 6)	The physical page holds a DEVIF metadata table
RESERVED	Other	N/A	Reserved

3.5.2. Physical Page Size

Three physical page size levels (4KB, 2MB and 1GB) are defined.

Table 3.16: Page Size Definition

Page Size	Associated Physical Page Size	Value
PS_1G	1GB	2
PS_2M	2MB	1
PS_4K	4KB	0

3.6. TD Private Memory Management Data Types: Secure EPT

Intel SDM, Vol. 3, 28.2.2 EPT Translation Mechanism

Note: This section describes private memory management types, as defined. Implementation may differ.

3.6.1. Secure EPT Levels

Secure EPT level definition is identical to legacy VMX EPT level definition. As a rule, an EPT entry at level L maps a GPA range whose size is 2^{12+9*L}.

Table 3.17: EPT Levels Definition

Level	0	1	2	3	4	5 (5-Level EPT Only)
GPA Range Size	4KB	2MB	1GB	512GB	256TB	16PB[2]
Child Physical Page Size	4KB	2MB	1GB	N/A	N/A	N/A
EPT Page Type	N/A	EPT	EPD	EPDPT	EPML4	EPML5
Parent EPT Entry Type	EPTE	EPDE	EPDPTE	EPML4E	EPML5E (5-level EPT) or VMCS.EPTP (4-level EPT)	VMCS.EPTP

[2] Only the lower half is available as TD private GPA space, because the SHARED bit must be 0

Level	0	1	2	3	4	5 (5-Level EPT Only)
GPA Offset Bits	20:12	29:21	38:30	47:39	51:48 (5-level EPT only)	N/A

3.6.2. Secure EPT Entry Information as Returned by TDX Module Functions

Many Intel TDX module functions return Secure EPT entry information. This information is returned in the formats detailed below, which may be different that the actual Secure EPT format as maintained by the TDX module in memory.

Note: The returned Secure EPT information is subject to change with new versions of TDX.

3.6.2.1. Returned L1 Secure EPT Entry Content

The returned L1 secure EPT entry format is detailed below. It may be different that the actual Secure EPT format as maintained by the TDX module in memory.

Table 3.18: L1 Secure EPT Entry Content as Returned by TDX Interface Functions

L1 Secure EPT Entry Field						Value Returned in RCX (per Entry State Returned in RDX)		
MSB	LSB	Size	Short Name	Full Name	Enabled	Non-FREE Leaf	Non-FREE Non-Leaf	FREE
0	0	1	R	Read	N/A	R	R	0
1	1	1	W	Write	N/A	W	W	0
2	2	1	X / Xs	Execute	N/A	X	X	0
5	3	3	MT	Memory Type	N/A	MT	0	0
6	6	1	IPAT	Ignore PAT	N/A	IPAT	0	0
7	7	1	PS	Leaf	N/A	1	0	0
8	8	1	A	Accessed	No	0	0	0
9	9	1	D	Dirty	No	0	0	0
10	10	1	Xu	Execute (User)	No	0	0	0
11	11	1	Ignored	Ignored	N/A	0	0	0
51	12	40	HPA[51:12]	Host Physical Address [51:12]	N/A	HPA[51:12]	HPA[51:12]	0
57	57	1	VGP	Verify Guest Paging	No	0	0	0
58	58	1	PWA	Paging-Write Access	No	0	0	0
59	59	1	Ignored	Ignored	N/A	0	0	0
60	60	1	SSS	Supervisor Shadow Stack	No	0	0	0
61	61	1	SPP	Check Sub-Page Permissions	No	0	0	0
62	62	1	Ignored	Ignored	N/A	0	0	0
63	63	1	SVE	Suppress #VE	Yes	SVE	0	1

For L1 SEPT entries, the R, W and X access permission bits' values depend on the SEPT entry state:

- For leaf entries in the MAPPED and EPORTED_DIRTY states, and non-leaf entries in the NL_MAPPED state, RWX = 111.
- For leaf entries in the BLOCKED, PENDING* and REMOVED states, non-leaf entries in the NL_BLOCKED state and FREE entries, RWX = 000.
- For leaf entries in the *BLOCKEDW* states, RWX = 101.

3.6.2.2. Returned L2 Secure EPT Entry Content

The returned L2 secure EPT entry format is detailed below. It may be different that the actual L2 Secure EPT format as maintained by the TDX module in memory.

Table 3.19: L2 Secure EPT Entry Content as Returned by TDX Interface Functions

L2 Secure EPT Entry Field						Value Returned in RCX (per Entry State Returned in RDX)		
MSB	**LSB**	**Size**	**Short Name**	**Full Name**	**Enabled**	**Non-FREE Leaf**	**Non-FREE Non-Leaf**	**FREE**
0	0	1	R	Read	N/A	R	R	0
1	1	1	W	Write	N/A	W	W	0
2	2	1	Xs	Execute	N/A	Xs	Xs	0
5	3	3	MT	Memory Type	N/A	MT	0	0
6	6	1	IPAT	Ignore PAT	N/A	IPAT	0	0
7	7	1	PS	Leaf	N/A	1	0	0
8	8	1	A	Accessed	No	0	0	0
9	9	1	D	Dirty	No	0	0	0
10	10	1	Xu	Execute (User)	No	Xu	Xu	0
11	11	1	Ignored	Ignored	N/A	0	0	0
51	12	40	HPA[51:12]	Host Physical Address [51:12]	N/A	HPA[51:12]	HPA[51:12]	0
57	57	1	VGP	Verify Guest Paging	No	0 / VGP	0	0
58	58	1	PWA	Paging-Write Access	No	0 / PWA	0	0
59	59	1	Ignored	Ignored	N/A	0	0	0
60	60	1	SSS	Supervisor Shadow Stack	No	0 / SSS	0	0
61	61	1	SPP	Check Sub-Page Permissions	No	0	0	0
62	62	1	Ignored	Ignored	N/A	0	0	0
63	63	1	SVE	Suppress #VE	Yes	SVE	0	1

For L2 SEPT entries, the R, W, Xs and Xu access permission bits' values depend on the L2 SEPT entry state and on the TD's ATTRIBUTE.DEBUG value:

- For leaf entries in the L2_MAPPED state:
  - If ATTRIBUTES.DEBUG is 0, then RWXsXu = 1111 and VGP, PWA and SSS are cleared to 0.
  - Else, the real values of RWXsXu and of VGP, PWA and SSS are returned.
- For leaf entries in the L2_BLOCKED state:
  - If ATTRIBUTES.DEBUG is 0, then RWXsXu = 0000 and VGP, PWA and SSS are cleared to 0.
  - Else, then RWXsXu = 0000 and GP, PWA and SSS are returned.
- For non-leaf entries in the L2_NL_MAPPED state, RWXsXu = 1111.
- For non-leaf entries in the L2_NL_BLOCKED state and L2_FREE entries, RWXsXu = 0000.

3.6.2.3. Additional Returned Secure EPT Information

Additional information for secure EPT entries is returned as defined below. Some SEPT entry state values are applicable only when enumerated by certain TDX_FEATURES bits (see 3.3.3.1). If a certain SEPT entry state value is not applicable, then it is considered reserved.

Table 3.20: Additional Secure EPT Entry Information Returned by TDX Interface Functions

Bits	Name	Description
2:0	Level	Level of the returned Secure EPT entry – see 3.6.1 above
7:3	Reserved	Set to 0
15:8	State	The TDX state of the Secure EPT entry – see Table 3.21 below
17:16	VM	Index of the VM for which the SEPT information is returned
63:18	Reserved	Set to 0

Table 3.21: Secure L1 EPT Entry TDX State Returned by TDX Interface Functions

L1 SEPT Entry State Name	Public State Number	TDX_FEATURES Enumeration	Description
FREE	0	N/A	L1 Secure EPT entry does not map a GPA range.
REMOVED	5	TD_MIGRATION or S4	L1 Secure EPT entry is of a removed page
NL_MAPPED	132	N/A	L1 Secure EPT entry maps a private GPA range which is accessible by the guest TD.
NL_BLOCKED	129	N/A	L1 Secure EPT entry maps a private GPA range, but new address translations to that range are blocked.
MAPPED	4	N/A	L1 Secure EPT entry maps a private GPA page which is accessible by the guest TD.
BLOCKED	1	N/A	L1 Secure EPT entry maps a private GPA page but new address translations to that range are blocked.
BLOCKEDW	8	TD_MIGRATION	L1 Secure EPT entry maps a private GPA page, but new address translations for write operations to that range are blocked.
EXPORTED_BLOCKEDW	9	TD_MIGRATION or S4	L1 Secure EPT entry maps a private page that has been blocked for writing and exported.
EXPORTED_DIRTY	11	TD_MIGRATION	L1 Secure EPT entry maps a private page that was exported but is not blocked for writing and its content and/or attributes may have since been modified.
EXPORTED_DIRTY_BLOCKEDW	12	TD_MIGRATION	L1 Secure EPT entry maps a private page that was previously exported, its content and/or attributes may have since been modified and then it was blocked for writing.
PENDING	2	N/A	L1 Secure EPT entry maps a 4KB or a 2MB page that has been dynamically added to the guest TD using TDH.MEM.PAGE.AUG and is pending acceptance by the guest TD using TDG.MEM.PAGE.ACCEPT. This page is not yet accessible by the guest TD.
PENDING_BLOCKED	3	N/A	L1 Secure EPT entry is both pending and blocked.
PENDING_BLOCKEDW	16	TD_MIGRATION	L1 Secure EPT entry is both pending and blocked for writing.
PENDING_EXPORTED_BLOCKEDW	17	TD_MIGRATION or S4	L1 Secure EPT entry is both pending and exported.

L1 SEPT Entry State Name	Public State Number	TDX_FEATURES Enumeration	Description
PENDING_EXPORTED_DIRTY	19	TD_MIGRATION	L1 Secure EPT entry is both pending and exported and is not blocked for writing.
PENDING_EXPORTED_DIRTY_BLOCKEDW	20	TD_MIGRATION	Secure EPT entry is both pending and exported and is blocked for writing.
MMIO_MAPPED	32	TDX_CONNECT	Secure EPT entry maps a private MMIO page which is accessible by the guest TD.
MMIO_BLOCKED	33	TDX_CONNECT	Secure EPT entry maps a private MMIO page, but new address translations to that page are blocked.
MMIO_PENDING	34	TDX_CONNECT	Secure EPT entry maps a 4KB, 2MB or 1GB MMIO page that is pending acceptance by the guest TD using TDG.MMIO.ACCEPT. This page is not yet accessible by the guest TD.
MMIO_PENDING_BLOCKED	35	TDX_CONNECT	Secure EPT entry for an MMIO page is both pending and blocked.

Table 3.22: Secure L2 EPT Entry TDX State Returned by TDX Interface Functions

L2 SEPT Entry State Name	Public State Number	TDX_FEATURES Enumeration	Description
L2_FREE	64	TD_PARTITIONING	L2 Secure EPT entry does not map a GPA range.
L2_NL_MAPPED	196	TD_PARTITIONING	L2 Secure EPT entry maps a private GPA range which is accessible by the L2 VM.
L2_NL_BLOCKED	193	TD_PARTITIONING	L2 Secure EPT entry maps a private GPA range, but new address translations to that range are blocked.
L2_MAPPED	68	TD_PARTITIONING	L2 Secure EPT entry maps a private GPA page which is accessible by the L2 VM.
L2_BLOCKED	65	TD_PARTITIONING	L2 Secure EPT entry maps a private GPA page but new address translations to that range are blocked.
L2_MMIO_MAPPED	96	TD_PARTITIONING and TDX_CONNECT	L2 Secure EPT entry maps a private MMIO page which is accessible by the L2 VM.
L2_MMIO_BLOCKED	97	TD_PARTITIONING and TDX_CONNECT	L2 Secure EPT entry maps a private MMIO page, but new address translations to that page are blocked.

3.6.3. GPA_ATTR: GPA Attributes

5 GPA_ATTR specifies the settable attributes of a page. It is used as an input of TDG.MEM.PAGE.ATTR.WR, as an output of TDG.PAGE.ATTR.WR, and for migration (TDH.EXPORT.MEM and TDH.IMPORT.MEM) and, if the TD is in debug mode, for returning of L2 attributes by TDH.MEM.SEPT.RD.

GPA_ATTR is an array of four GPA_ATTR_SINGLE_VM 16-bit entries:

Table 3.23: GPA_ATTR: GPA Attributes (all VMs) Definition

Bits	VM Index	Bits	Description
15:0	0	15:0	GPA attributes for L1 (all-0 for migration)
31:16	1	31:24	GPA attributes for L2 VM #1
47:32	2	47:32	GPA attributes for L2 VM #2

Bits	VM Index	Bits	Description
63:48	3	63:48	GPA attributes for L2 VM #3

Table 3.24: GPA_ATTR_SINGLE_VM: GPA Attributes (Single VM) Definition

Bit(s)	Size	Attribute Type	Name	Description	TDG.MEM.PAGE. ATTR.RD/WR Access			Used for Migration	
					L1	L2 Mem.	L2 MMIO[3]	L1	L2
0	1	Intel64	R	Read	None	RW	RW	No	Yes
1	1	Intel64	W	Write	None	RW	RW	No	Yes
2	1	Intel64	Xs	Execute (Supervisor)	None	RW	R	No	Yes
3	1	Intel64	Xu	Execute (User)	None	RW	R	No	Yes
4	1	Intel64	VGP	Verify Guest Paging	None	RW	R	No	Yes
5	1	Intel64	PWA	Paging-Write Access	None	RW	R	No	Yes
6	1	Intel64	SSS	Supervisor Shadow Stack	None	RW	R	No	Yes
7	1	Intel64	RESERVED	Reserved, must be 0	None	None	None	No	No
14:8	7	N/A	RESERVED	Reserved, must be 0	None	R	R	No	No
15	1	TDX	VALID	Indicates that the other bits are valid. If its value is 0, other fields are reserved and must be 0.	RW	RW	RW	No	Yes

3.6.3.1. GPA Attributes Rules

The TDX module enforces the following rules to help ensure that GPA attributes will not cause an EPT Misconfiguration (see [Intel SDM, Vol. 3, 28.3.3.1]):

- If VALID is 0, all other bits must be 0
- Reserved bits must be 0.
- If bit W is 1, bit R must be 1
- If bit PWA is 1, bit R must be 1 (regardless of the VMCS “EPT paging-write control” VM-execution control.

The TDX module checks, on TDH.SYS.INIT, that the CPU supports setting Xs or Xu when R is 0.

3.6.4. GLA List

GLA lists are used by TDG.VP.INVGLA.

3.6.4.1. GLA_LIST_ENTRY

GLA_LIST_ENTRY species a range of consecutive guest linear addresses, each aligned on 4KB.

Table 3.25: GLA_LIST_ENTRY Definition

Bits	Name	Description
11:0	**LAST_GLA_INDEX**	Index of the last 4KB-aligned linear address to be processed
63:12	**BASE_GLA**	Bits 63:12 of the guest linear address of the first 4KB page to be processed

[3] Applicable only if the TDX module supports TDX Connect

3.6.4.2. GLA_LIST

A GLA_LIST is an array of up to 512 GLA_LIST_ENTRIES.

3.6.4.3. GLA_LIST_INFO: GLA List GPA and Additional Information

GLA_LIST_INFO is a 64b structure used as a GPR input and output operand of TDG.VP.INVGLA. It provides the GPA of the GLA list page in private memory, the index of the first entry and the number of entries to be processed.

Table 3.26: GLA_LIST_INFO

Bits	Name	Description
8:0	**FIRST_ENTRY**	Index of the first entry of the list to be processed
11:9	**RESERVED**	Reserved: must be 0
51:12	**LIST_GPA**	Bits 51:12 of the guest physical address of the GLA list page, which must be a private GPA
61:52	**NUM_ENTRIES**	Number of entries in the GLA list to be processed, must be between 0 through 512
63:62	**RESERVED**	Reserved: must be 0

3.7. TD Entry and Exit Types

3.7.1. Extended Exit Qualification

Extended Exit Qualification is a 64-bit field returned by TDH.VP.ENTER for asynchronous TD exits with an architectural VMX exit reasons. It contains additional non-VMX, TDX-specific information.

Table 3.27: Extended Exit Qualification

Bits	Name	Description		
3:0	TYPE	Extended exit qualification type		
		Value	**Name**	**Description**
		0	NONE	No extended exit qualification
		1	ACCEPT	Extended exit qualification for an EPT violation during TDG.MEM.PAGE.ACCEPT
		2	GPA_DETAILS	Extended exit qualification for an EPT violation caused by guest-side interface function failure of GPA→HPA translation
		3	TD_ENTRY_MSR_LOAD_FAILURE	Extended exit qualification for failures of TD entry due to loading guest MSR state
		4	TD_ENTRY_XSTATE_LOAD_FAILURE	Extended exit qualification for failures of TD entry due to loading guest extended state
		5	ATTR_WR	Extended exit qualification for an EPT violation during TDG.MEM.PAGE.ATTR.WR

Bits	Name	Description		
		6	PENDING_EPT_VIOLATION[4]	Extended exit qualification for an EPT violation due to guest TD access to a PENDING page
		Other	Reserved	
31:4	Reserved	Set to 0		
63:32	INFO	TYPE-specific information		

TYPE	Value
NONE	0
ACCEPT	See the table below
GPA_DETAILS	See the table below
TD_ENTRY_MSR_LOAD_FAILURE	MSR index
TD_ENTRY_XSTATE_LOAD_FAILURE	0
ATTR_WR	See the table below
PENDING_EPT_VIOLATION	0
Reserved	0

Table 3.28: Extended Exit Qualification INFO Field (Bits 63:32) when TYPE is ACCEPT or ATTR_WR

Bits	Name	Description
34:32	REQ_SEPT_LEVEL	SEPT level requested as an input to TDG.MEM.PAGE.ACCEPT or TDG.MEM.PAGE.ATTR.WR
37:35	ERR_SEPT_LEVEL	SEPT level in which TDG.MEM.PAGE.ACCEPT or TDG.MEM.PAGE.ATTR.WR detected an error
45:38	ERR_SEPT_STATE	The TDX state of the Secure EPT entry where TDG.MEM.PAGE.ACCEPT or TDG.MEM.PAGE.ATTR.WR detected an error – see Table 3.21 above
46	ERR_SEPT_IS_LEAF	Indicates that the SEPT entry where TDG.MEM.PAGE.ACCEPT or TDG.MEM.PAGE.ATTR.WR detected an error is a leaf entry
63:47	Reserved	Set to 0

Table 3.29: Extended Exit Qualification INFO Field (Bits 63:32) when TYPE is GPA_DETAILS

Bits	Name	Description
34:32	Reserved	Set to 0
37:35	ERR_SEPT_LEVEL	Level where the Secure EPT walk error occurred
51:38	Reserved	Set to 0
53:52	VM_INDEX	Virtual machine index for which Secure EPT walk error occurred
63:54	Reserved	Set to 0

[4] Availability of this indication is enumerated by TDX_FEATURES0.PENDING_EPT_VIOLATION_V2 (bit 16), readable by TDH.SYS.RD*.

3.8. L2 VM Transition Types

3.8.1. L2_ENTER_GUEST_STATE

L2_ENTER_GUEST_STATE is used as input and output of TDG.VP.ENTER. It is an array of general-purpose (GPR) register values, organized according to their architectural number, with additional values of RFLAG, RIP and SSP.

Table 3.30: L2_ENTER_GUEST_STATE Definition

Offset (Bytes)	Size (Bytes)	Name	Description
0	8	RAX	
8	8	RCX	
16	8	RDX	
24	8	RBX	
32	8	RSP	
40	8	RBP	
48	8	RSI	
56	8	RDI	
64	8	R8	
72	8	R9	
80	8	R10	
88	8	R11	
96	8	R12	
104	8	R13	
112	8	R14	
120	8	R15	
128	8	RFLAGS	
136	8	RIP	
144	8	SSP	
152	2	GUEST_INTERRUPT_STATUS	Bits 7:0: RVI Bits 15:7: SVI

3.9. Measurement and Attestation Types

Note: This section describes measurement and attestation types, as defined. Implementation may differ.

3.9.1. CPUSVN

CPUSVN is a 16B Security Version Number of the CPU.

- There is a single CPUSVN used for SGX and TDX.
- CPUSVN contents are considered micro-architectural. CPUSVN is composed of fields for PR_RESET_SVN, R_LAST_PATCH_SVN, SINIT, BIOS ACM, Boot Guard ACM and BIOS Guard NP-PPPE module.

3.9.2. TDREPORT_STRUCT

TDREPORT_STRUCT is the output of the TDG.MR.REPORT function. TDREPORT_STRUCT is composed of a generic MAC structure (REPORTMACSTRUCT, see 3.9.5 below), a TEE_TCB_INFO structure and a TDX-specific TEE info structure (TDINFO_STRUCT, see 3.9.7 below).

The overall size of TDREPORT_STRUCT depends on its version, as specified in REPORTMACSTRUCT.REPORTTYPE.VERSION:

- For REPORTTYPE.VERSION values of 0 and 1, TDREPORT_STRUCT's size is 1024 bytes.

Table 3.31: TDREPORT_STRUCT Definition

Name	Offset (Bytes)	Type	Size (Bytes)	Description
REPORTMACSTRUCT	0	REPORTMACSTRUCT	256	REPORTMACSTRUCT for the TDG.MR.REPORT
TEE_TCB_INFO	256	TEE_TCB_INFO_STRUCT	239	Additional attestable elements in the TD's TCB are not reflected in the REPORTMACSTRUCT.CPUSVN – includes the Intel TDX module measurements.
RESERVED	495	N/A	17	Reserved – contains 0
TDINFO	512	TDINFO_STRUCT	See 3.9.7	Structure containing the TD's attestable properties. • The hash of this structure is found in REPORTMACSTRUCT.TEE_INFO_HASH. • Size is determined by REPORTMACSTRUCT.REPORTTYPE.VERSION. See 3.9.7 for details.

3.9.3. TEE_TCB_INFO (Reference)

TEE_TCB_INFO is defined in the [TDX Arch Extensions Spec]. The definition below is provided for reference. Some details which are not applicable for TDX have been eliminated.

The size of TEE_TCB_INFO is 239 bytes.

Table 3.32: TEE_TCB_INFO Definition

Name	Offset (Bytes)	Size (Bytes)	Description
VALID	0	8	Indicates which TEE_TCB_INFO fields are valid. • 1 in the i^{th} significant bit reflects that the 8 bytes starting at offset (8 * i) are valid • 0 in the i^{th} significant bit reflects that either 8 bytes starting at offset (8 * i) is not populated or reserved and is set to zero. Set to 0x301FF.
TEE_TCB_SVN	8	16	TEE_TCB_SVN of the TDX module that created the TD on the current platform. **TD Migration:** For a TD which has been migrated, this is the TEE_TCB_SVN of the TDX module on the destination platform, at the time of destination TD creation (TDH.MNG.CREATE), before import.
MRSEAM	24	48	The measurement of the TDX module that created the TD on the current platform. **TD Migration:** For a TD which has been migrated, this is the measurement of the TDX module on the destination platform, at the time of destination TD creation (TDH.MNG.CREATE), before import.

Name	Offset (Bytes)	Size (Bytes)	Description
MRSIGNERSEAM	72	48	Set to all 0's.
ATTRIBUTES	120	8	Set to all 0's.
TEE_TCB_SVN2	128	16	TEE_TCB_SVN of the current TDX module on the current platform. **TD Migration:** For a TD which has been migrated, this is the measurement of the current TDX module on the destination platform, at the time TDREPORT_STRUCT is generated by TDG.MR.REPORT. **Note:** TEE_TCB_SVN2 may be different that TEE_TCB_SVN, due to TD-preserving TDX module updates.
RESERVED	144	95	Set to all 0's.

3.9.4. TEE_TCB_SVN (Reference)

TEE_TCB_SVN is defined in the [TDX Arch Extensions Spec]. The definition below is provided for reference.

Name	Offset (Bytes)	Size (Bytes)	Description
TDX_MODULE_SVN_MINOR	0	1	TDX module minor SVN
TDX_MODULE_SVN_MAJOR	1	1	TDX module major SVN
SEAM_LAST_PATCH_SVN	2	1	Microcode SE_SVN at the time the TDX module was loaded
RESERVED	3	13	Must be Zero

3.9.5. REPORTMACSTRUCT (Reference)

Note: REPORTMACSTRUCT is defined in the [TDX Arch Extensions Spec]; the definition below is provided for reference.

REPORTMACSTRUCT is the first field in the TEE report structure. It is common to Intel's Trusted Execution Environments (TEEs) – e.g., SGX and TDX. In the TDX architecture, that is TDREPORT_STRUCT. REPORTMACSTRUCT is MAC-protected and contains hashes of the remainder of the report structure which includes the TEE's measurements, and where applicable, the measurements of additional TCB elements not reflected in REPORTMACSTRUCT.CPUSVN – e.g., a SEAM's measurements.

Software verifying a TEE report structure (for TDX, this includes TEE_TCB_INFO_STRUCT and TDINFO_STRUCT) should check the following:

1. Check that REPORTMACSTRUCT.TEE_INFO_HASH equals SHA384(TDINFO_STRUCT).
2. If REPORTMACSTRUCT.TEE_TCB_INFO_HASH is not 0, check that REPORTMACSTRUCT.TEE_TCB_INFO_HASH equals SHA384(TEE_TCB_INFO).

If all checks pass, the measurements in the structure describe a TEE on this platform.

The size of REPORTMACSTRUCT is 256B.

Table 3.33: REPORTMACSTRUCT Definition

Name	Offset (Bytes)	Type	Size (Bytes)	Description	MAC
REPORTTYPE	0	REPORTTYPE	4	Type Header Structure	Yes
RESERVED	4		12	Must be zero	Yes

Name	Offset (Bytes)	Type	Size (Bytes)	Description	MAC
CPUSVN	16	CPUSVN	16	CPU SVN	Yes
TEE_TCB_INFO_HASH	32	SHA384_HASH	48	For TDX, SHA384 of TEE_TCB_INFO	Yes
TEE_INFO_HASH	80	SHA384_HASH	48	SHA384 of TEE_INFO: a TEE-specific info structure (TDINFO_STRUCT or SGXINFO) or 0 if no TEE is represented	Yes
REPORTDATA	128		64	A set of data used for communication between the caller and the target.	Yes
RESERVED	192		32	Must be zero	Yes
MAC	224		32	The MAC over the REPORTMACSTRUCT with model-specific MAC	No

3.9.6. REPORTTYPE (Reference)

Note: REPORTTYPE is defined in the [TDX Arch Extensions Spec]; the definition below is provided for reference.

REPORTTYPE indicates the reported Trusted Execution Environment (TEE) type, sub-type and version.

The size of REPORTTYPE is 4B.

Table 3.34: TDX-Specific REPORTTYPE Definition

Name	Offset (Bytes)	Size (Bytes)	Description	Value
TYPE	0	1	Trusted Execution Environment (TEE) Type	0x00: SGX 0x7F-0x01: Reserved (TEE implemented by CPU) 0x80: Reserved (TEE implemented by a SEAM module) **0x81: TDX** 0xFF-0x82: Reserved (TEE implemented by a SEAM module)
SUBTYPE	1	1	TYPE-specific subtype	0: Standard TDX report Other: Reserved
VERSION	2	1	TYPE-specific version.	For TDX, VERSION may have the following values: 0: There are no bound nor pre-bound service TDs. TDINFO_STRUCT.SERVTD_HASH is not used (its value is 0). 1: TDINFO_STRUCT.SERVTD_HASH is used.
RESERVED	3	1	Must be zero	0

3.9.7. TDINFO_STRUCT

TDINFO_STRUCT is defined as the TDX-specific TEE_INFO part of TDG.MR.REPORT. It contains the measurements and initial configuration of the TD that was locked at initialization and a set of measurement registers that are run-time extendable. These values are copied from the TDCS by the TDG.MR.REPORT function. Refer to the [TDX Module Base Spec] for additional details.

TDINFO_STRUCT is composed of a base set of fields and an extension. The content of the extension and the overall size of TDINFO_STRUCT depend on the report version, as specified in REPORTMACSTRUCT.REPORTTYPE.VERSION:

- For REPORTTYPE.VERSION values of 0 and 1, TDREPORT_STRUCT's size is 512 bytes.

Table 3.35: Overall TDINFO_STRUCT Definition

Name	Offset (Bytes)	Size (Bytes)		Description
TDINFO_BASE	0	448		Base TDINFO fields
TDINFO_EXTENSION	448	The extension field depends on REPORTMACSTRUCT.REPORTTYPE.VERSION:		
		REPORTTYPE.VERSION	**Size**	**Description**
		0, 1	64	Reserved, must be zero

TDINFO_BASE

Table 3.36: TDINFO_BASE Definition

Name	Offset (Bytes)	Type	Size (Bytes)	Description
ATTRIBUTES	0		8	TD's ATTRIBUTES
XFAM	8		8	TD's XFAM
MRTD	16	SHA384_HASH	48	Measurement of the initial contents of the TD
MRCONFIGID	64	SHA384_HASH	48	Software-defined ID for non-owner-defined configuration of the guest TD – e.g., run-time or OS configuration
MROWNER	112	SHA384_HASH	48	Software-defined ID for the guest TD's owner
MROWNERCONFIG	160	SHA384_HASH	48	Software-defined ID for owner-defined configuration of the guest TD – e.g., specific to the workload rather than the run-time or OS
RTMR	208	SHA384_HASH	4 * 48	Array of 4 run-time extendable measurement registers
SERVTD_HASH	400	SHA384_HASH	48	If there is one or more bound or pre-bound service TDs, SERVTD_HASH is the SHA384 hash of the TDINFO_STRUCTs of those service TDs bound. Else, SERVTD_HASH is 0.

3.10. Metadata Access Types

Note: This section describes control structure field access types, as defined. Implementation may differ.

Metadata access is described in the [TDX Module Base Spec].

3.10.1. MD_FIELD_ID: Metadata Field Identifier / Sequence Header

MD_FIELD_ID is used for two purposes:

Metadata Field Identifier: Used for specifying a single element of a metadata field

Metadata Sequence Header: Used as the header of a metadata field sequence

Lists of metadata field identifiers for global-scope metadata, TD-scope metadata and VCPU-scope metadata are provided in Ch. 4. The metadata tables provide a **base identifier**. The table below specifies which components of MD_FIELD_ID are taken from the base identifier; other components need to be specified as required.

Table 3.37: MD_FIELD_ID (Metadata Field Identifier / Sequence Header) Definition

Bits	Size	Name	From Metadata Tables' Base Field ID	Single-Element or Sequence Header	Description
23:0	24	**FIELD_CODE**	Yes	Both	For a single-element identifier, identifies the element that is being accessed. For a metadata sequence header, identifies the first field that is being accessed in a sequence.
31:24	8	**RESERVED**	No	Both	Must be 0
33:32	2	**ELEMENT_SIZE_CODE**	Yes	Both	Size of a single element of a metadata field: **0:** 8 bits **1:** 16 bits **2:** 32 bits **3:** 64 bits For backward compatibility, TDH.MNG.RD, TDH.MNG.WR, TDH.VP.RD and TDH.VP.WR version 0 ignore this field and use a default value based on the field code.
37:34	4	**LAST_ELEMENT_ IN_FIELD**	No	Sequence Header	Number of elements in a metadata field, minus 1 For a single-element identifier, the value is 0. This field is ignored when used as input to TD*.SYS.RDALL.
46:38	9	**LAST_FIELD_ IN_SEQUENCE**	No	Sequence Header	Number of fields in a sequence, minus 1 For a single-element identifier, the value is 0. This field is ignored when used as input to TD*.SYS.RDALL.
49:47	3	**RESERVED**	Yes	Both	Must be 0
50	1	**INC_SIZE**	Yes	Sequence Header	For a single-element identifier, INC_SIZE is ignored. For a sequence header, INC_SIZE specifies how FIELD_CODE is incremented when accessing consecutive elements in a sequence: **0:** Increment FIELD_CODE by 1 for each element. **1:** Increment FIELD_CODE by 2 for each element. INC_SIZE is designed to support VMCS field encoding, where bit 0 (access type) is always 0 for full access.
51	1	**WRITE_MASK_VALID**	No	Both	Indicates that a write mask is provided together with the write value. For backward compatibility, single-element metadata write interface functions (e.g., TDH.MNG.WR, TDH.VP.WR etc.) and use an implicit value of 1. This field is ignored by metadata read interface functions.

Bits	Size	Name	From Metadata Tables' Base Field ID	Single-Element or Sequence Header	Description
54:52	3	**CONTEXT_CODE**	Yes	Both	Specifies the context of the field: **0:** Platform (whole Intel TDX module) **1:** TD **2:** TD VCPU **Other:** Reserved All metadata read and write interface functions (e.g., TDH.MNG.RD, TDH.MNG.WR, TDH.VP.RD, TDG.SYS.RDALL etc.) ignore this field when used as an input; they use an implicit value.
55	1	**RESERVED**	Yes	Both	Must be 0
61:56	6	**CLASS_CODE**	Yes	Both	Identifies the class of the fields being accessed Class codes are defined in 3.10.3.
62	1	**RESERVED**	Yes	Both	Must be 0
63	1	**NON_ARCH**	Yes	Both	Specifies forward compatibility, i.e., whether this field identifier will maintain their definition in a compatible way throughout Intel TDX module updates. 0: Field identifier will maintain forward compatibility. 1: Field identifier may not maintain forward compatibility. **Note:** Even if the NON_ARCH bit value is 1, identifiers of migratable fields will in most cases maintain forward compatibility, to support TD migration between different TDX module releases.

Values Reserved for Software Use

Bits 63:52 value of all-1 will never be used by the TDX module. This range is reserved for use by host VMM and guest TD software.

3.10.2. Meaning of Field Codes

For some field classes, field codes have an architectural meaning, as shown below. For other classes, field codes are arbitrarily assigned.

Table 3.38: Meaning of Field Codes

Field Class	Field Code Meaning			Reference
VMCS	Field code is the architectural VMCS field code. The "HIGH" access type (for accessing the upper 32b of 64b fields) is not supported.			[Intel SDM, Vol. 3, 24.11.2 and App. B]
	Bits	**Name**	**Description**	
	23:16	RESERVED	Must be 0	
	15:0	VMCS_FIELD_CODE	Bits 15:0 of the architectural VMCS field code **Note:** Bits 32:16 of the VMCS field code are implicitly 0.	
MSR Bitmap	Offset (in 8B units) from the beginning of the architectural MSR bitmaps page			[Intel SDM, Vol. 3, 24.6.9]
Secure EPT Root	Offset (in 8B units) from the beginning of the page			

<table>
<tr><th>Field Class</th><th>Field Code Meaning</th><th>Reference</th></tr>
<tr><td>Virtual APIC Page</td><td>Offset (in 8B units) from the beginning of the architectural virtual APIC page structure</td><td>[Intel SDM, Vol. 3, 29.1]</td></tr>
<tr><td>CPUID Config</td><td>Each field contains two 64-bit element, with the values returned by CPUID for the leaf and sub-leaf, as follows:
Element 0[31:0]: EAX Element 0[63:32]: EBX
Element 1[31:0]: ECX Element 1[63:32]: EDX
The field code is packed as shown below:
<table>
<tr><th>Bits</th><th>Name</th><th>Description</th></tr>
<tr><td>23:17</td><td>RESERVED</td><td>Must be 0</td></tr>
<tr><td>16</td><td>LEAF_31</td><td>Leaf number bit 31</td></tr>
<tr><td>15:9</td><td>LEAF_6_0</td><td>Leaf number bit 6:0
Note: Leaf bits 30:7 are implicitly 0.</td></tr>
<tr><td>8</td><td>SUBLEAF_NA</td><td>Sub-leaf not applicable flag</td></tr>
<tr><td>7:1</td><td>SUBLEAF_6_0</td><td>Sub-leaf number bits 6:0
If SUBLEAF_NA is 1, then SUBLEAF_6_0 is all-1.
Note: Sub-leaf bits 31:7 are implicitly 0.</td></tr>
<tr><td>0</td><td>ELEMENT_I</td><td>Element index within field</td></tr>
</table></td><td></td></tr>
<tr><td>GPR State</td><td>Architectural GPR number</td><td></td></tr>
<tr><td>MSR State</td><td>Architectural MSR index, packed as shown below:
<table>
<tr><th>Bits</th><th>Description</th></tr>
<tr><td>23:14</td><td>Reserved, must be 0</td></tr>
<tr><td>13</td><td>Bit 31 (equal to bit 30) of the architectural MSR index</td></tr>
<tr><td>12:0</td><td>Bits 12:0 of the architectural MSR index</td></tr>
</table></td><td></td></tr>
<tr><td>Extended State</td><td>Offset (in 8B units) from the beginning of the page extended state buffer</td><td></td></tr>
<tr><td>Other</td><td>Arbitrary field identifiers</td><td></td></tr>
</table>

3.10.3. Class Codes

3.10.3.1. TDX Module Global Scope Field Class Codes

TDX Module global scope field classes are defined as follows:

Table 3.39: TDX Module Global Scope Field Class Codes Definition

Class Code	Field Class Name
0	Platform Info
8	TDX Module Version
9	TDX Module Handoff
10	TDX Module Info
16	CMR Info
17	TDMR Info

Class Code	Field Class Name
24	TD Control Structures
25	TD Configurability
26	Memory Management
27	Measurement
32	Migration
33	Service TD
34	TD Partitioning
48	TDX Connect

3.10.3.2. TD-Scope (TDR and TDCS) Field Class Codes

TD-scope field classes are defined as follows:

Table 3.40: TD Scope (TDR and TDCS) Field Class Codes Definition

Class Code	Control Structure	Field Class Name
0	TDR	TD Management
1	TDR	Key Management
2	TDR	TD Preserving
3	TDR	TDX I/O
16	TDCS	TD Management
17	TDCS	Execution Controls
18	TDCS	TLB Epoch Tracking
19	TDCS	Measurement
20	TDCS	CPUID
21	TDCS	Zero Page
22	TDCS	Virt. MSR Values
24	TDCS	Migration
25	TDCS	Service TD
26	TDCS	MIGSC Links
27	TDCS	TDX I/O
32	TDCS	MSR Bitmaps
33	TDCS	Secure EPT Root
37	TDCS	L2 Secure EPT Root [1]
41	TDCS	L2 Secure EPT Root [2]
45	TDCS	L2 Secure EPT Root [3]

3.10.3.3. VCPU-Scope (TDVPS) Field Class Codes

TDVPS field classes are defined as follows:

Table 3.41: TD VCPU Scope (TDVPS) Field Class Codes Definition

Class Code	Field Class Name
0	TD VMCS
1	VAPIC
2	VE_INFO
16	Guest GPR State
17	Guest State
18	Guest Ext. State
19	Guest MSR State
32	Management
33	CPUID Control
34	EPT Violation Log
36	VMCS[1]
37	MSR Bitmaps[1]
38	MSR Bitmaps Shadow[1]
44	VMCS[2]
45	MSR Bitmaps[2]
46	MSR Bitmaps Shadow[2]
52	VMCS[3]

3.10.4. Order of Field Identifiers

For usages such as TD migration, there is a need to define strict ordering between field identifiers. For this purpose, we consider field identifiers to be orders by the following fields:

1. CONTEXT_CODE
2. CLASS_CODE
3. FIELD_CODE

3.10.5. MD_LIST_HEADER: Metadata List Header

MD_LIST_HEADER is defined below. The size of MD_LIST_HEADER is 64 bits.

Table 3.42: MD_LIST_HEADER Definition

Bits	Name	Description
15:0	LIST_BUFF_SIZE	The size of memory buffer containing the list The buffer may be larger than the actual space occupied by the list; in this case the excess buffer space is ignored or read and may be overwritten on write.
31:16	NUM_SEQUENCES	The number of metadata field sequences in the list.
63:32	RESERVED	Reserved, set to 0

3.10.6. Private Page List

A private page list specifies a list of HPAs of 4KB pages that are, or will become, TD private pages. The list may have up to 512 64-bit entries, each containing a 4KB-aligned HPA (HKID bits must be 0) of a page. The list is contained in a single 4KB page and must be aligned on 4KB. The page list may contain null entries, indicated by the INVALID bit.

Table 3.43: Private Page List Entry

Bits	Name	Description
11:0	**RESERVED**	Reserved: must be 0 (unless bit 63 indicates an invalid entry)
51:12	**HPA**	Bits 51:12 of the host physical address (HKID bits must be 0) of the migration buffer page
62:52	**RESERVED**	Reserved: must be 0 (unless bit 63 indicates an invalid entry)
63	**INVALID**	A value of 1 indicates that this entry is invalid

3.10.7. HPA_AND_SIZE: HPA and Size of a Buffer

HPA_AND_SIZE is a 64-bit structure used to provide a buffer host physical address and size details.

Table 3.44: HPA_AND_SIZE

Bits	Name	Description
51:0	**HPA**	Bits 51:0 of the host physical address (including HKID bits) of the buffer
63:52	**SIZE**	Size of the buffer, in bytes

3.10.8. HPA_AND_LAST: HPA and Last Byte Index of a Page-Aligned Buffer

HPA_AND_LAST is a 64-bit structure used to provide a 4KB aligned buffer host physical address and size details.

Table 3.45: HPA_AND_LAST

Bits	Name	Description
11:0	**LAST**	Index of the last byte in the buffer
51:12	**HPA**	Bits 51:12 of the host physical address (including HKID bits) of the 4KB-aligned buffer
63:52	**RESERVED**	Reserved: must be 0

3.11. Service TD Types

3.11.1. SERVTD_BINDING_TABLE: Service TD Binding Table

SERVTD_BINDING_TABLE is a table of service TD binding information, held in the TDCS. For details, see the [TDX Module Base Spec].

Table 3.46: Service TD Binding Entry Definition

Field	Type	Offset (Bytes)	Size (Bytes)	Description
STATE	SERVTD_BINDING_STATE	0	1	See below and [TDX Module Base Spec]
Reserved		1	1	Must be 0

Field	Type	Offset (Bytes)	Size (Bytes)	Description
TYPE	SERVTD_TYPE	2	2	See below and [TDX Module Base Spec]
Reserved		4	4	Must be 0
ATTR	SERVTD_ATTR	8	8	See below and [TDX Module Base Spec]
UUID	256-bit blob	16	32	See [TDX Module Base Spec]
INFO_HASH	SHA384_HASH	48	48	See [TDX Module Base Spec]
Reserved		96	32	Must be 0

TD-Preserving Update TDX Module Handoff Compatibility

SERVTD_BINDING_TABLE is preserved in memory across TD-preserving updates. The table below specifies the MODULE_HV versions for which the above MIGSC definition is appliable.

Table 3.47: SERVTD_BINDING_TABLE Compatibility with TD Preserving Updates

Module Handoff Version	Value
Minimum MODULE_HV	0
Maximum MODULE_HV	0

3.11.2. SERVTD_BINDING_STATE: Service TD Binding State

SERVTD_BINDING_STATE indicates the state of the service TD binding slot. For details, see the [TDX Module Base Spec].

Table 3.48: SERVTD_BINDING_STATE Values

Value	Name
0	NOT_BOUND
1	PRE_BOUND
2	BOUND

3.11.3. SERVTD_TYPE: Service TD Binding Type

SERVTD_TYPE is a 16-bit field which specifies the binding type of a service TD. For details, see the [TDX Module Base Spec].

Table 3.49: SERVTD_TYPE Definition

Value	Meaning	Multiple Bindings	Metadata Access
0	Migration TD	No	Migration session key
Other	Reserved	N/A	N/A

3.11.4. SERVTD_ATTR: Service TD Binging Attributes

SERVTD_ATTR is a 64-bit field which specifies binding attributes of a service TD. For details, see the [TDX Module Base Spec].

Table 3.50: SERVTD_ATTR Definition

Bit(s)	Name	Description
31:0	RESERVED	Must be 0

Bit(s)	Name	Description
32	IGNORE_ATTRIBUTES	If set to 1, a value of 0 is used instead of the service TD's ATTRIBUTES field when calculating SERVTD_INFO_HASH
33	IGNORE_XFAM	If set to 1, a value of 0 is used instead of the service TD's XFAM field when calculating SERVTD_INFO_HASH
34	IGNORE_MRTD	If set to 1, a value of 0 is used instead of the service TD's MRTD field when calculating SERVTD_INFO_HASH
35	IGNORE_MRCONFIGID	If set to 1, a value of 0 is used instead of the service TD's MRCONFIGID field when calculating SERVTD_INFO_HASH
36	IGNORE_MROWNER	If set to 1, a value of 0 is used instead of the service TD's MROWNER field when calculating SERVTD_INFO_HASH
37	IGNORE_MROWNERCONFIG	If set to 1, a value of 0 is used instead of the service TD's MROWNERCONFIG field when calculating SERVTD_INFO_HASH
38	IGNORE_RTMR0	If set to 1, a value of 0 is used instead of the service TD's RTMR0 field when calculating SERVTD_INFO_HASH
39	IGNORE_RTMR1	If set to 1, a value of 0 is used instead of the service TD's RTMR1 field when calculating SERVTD_INFO_HASH
40	IGNORE_RTMR2	If set to 1, a value of 0 is used instead of the service TD's RTMR2 field when calculating SERVTD_INFO_HASH
41	IGNORE_RTMR3	If set to 1, a value of 0 is used instead of the service TD's RTMR3 field when calculating SERVTD_INFO_HASH
42	IGNORE_SERVTD_HASH	If set to 1, a value of 0 is used instead of the service TD's SERVTD_HASH field when calculating SERVTD_INFO_HASH
Other	RESERVED	Must be 0

3.12. Migration Types

Enumeration: The following definitions are applicable for TDX modules which support TD migration, as enumerated by TDX_FEATURES0.TD_MIGRATION (bit 0) or S4, as enumerated by TDX_FEATURES0.S4 (bit 13).

3.12.1. MBMD: Migration Bundle Metadata

MBMD is composed of a common header and a variable type-specific information.

3.12.1.1. Generic MBMD Structure

The maximum overall size of MBMD is 128 bytes.

Table 3.51: Generic MBMD Structure Definition

Field	Offset (Bytes)	Size (Bytes)	Description	Included In MAC	Included in IV
SIZE	0	2	Overall size of the MBMD structure, in bytes	Yes	No
MIG_VERSION	2	2	Migration protocol version Changes in MBMB format, other migration bundle components format or migration protocol sequence require updating the protocol version. Migration protocol version is set by the MigTD before migration session starts.	Yes	No

Field	Offset (Bytes)	Size (Bytes)	Description	Included In MAC	Included in IV
MIGS_INDEX	4	2	Index of the migration stream used for migrating this migration bundle	As 0	Yes
MB_TYPE	6	1	The type of information being migrated: 0: TD-scope immutable non-memory state 1: TD-scope mutable non-memory state 2: VCPU-scope mutable non-memory state 3–15: Reserved 16: TD private memory 17–31: Reserved 32: Epoch token 33: Abort token Other: Reserved	Yes	No
RESERVED	7	1	Reserved, must be 0	Yes	No
MB_COUNTER	8	4	Per-stream migration bundle counter Starts from 0 on each migration epoch start, incremented by 1 on each MBMD export to the associated stream.	Yes	No
MIG_EPOCH	12	4	Migration epoch Starts from 0 on migration session start, incremented by 1 on each epoch token. A value of 0xFFFFFFFF indicates out-of-order phase.	Yes	No
IV_COUNTER	16	8	Monotonously incrementing counter, used as a component in the AES-GCM IV	As 0	Yes
Type-Specific Information	24	Variable	Variable-sized additional information for each specific type of MBMD	Yes	No
MAC	24+V	16	AES-256-GCM MAC over other MBMD fields and any associated migration data (all the migration pages)	No	No

3.12.1.2. TD-Scope Immutable Non-Memory State MBMD Fields

Table 3.52: TD-Scope Immutable Non-Memory State MBMD Fields Definition

Field	Offset (Bytes)	Size (Bytes)	Description	Included In MAC	Included in IV
NUM_F_MIGS	24	2	Maximum number of forward migration streams that will be used	Yes	No
RESERVED	26	2	Reserved, must be 0	Yes	No
NUM_SYS_MD_PAGES	28	1	Number of pages in the page list used for migrating TDX module metadata	Yes	No
RESERVED	29	3	Reserved, must be 0	Yes	No

3.12.1.3. TD-Scope Mutable Non-Memory State MBMD Fields

Table 3.53: TD-Scope Mutable Non-Memory State MBMD Fields Definition

Field	Offset (Bytes)	Size (Bytes)	Description	Included In MAC	Included in IV
RESERVED	24	8	Reserved, must be 0	Yes	No

3.12.1.4. VCPU-Scope Mutable Non-Memory State MBMD Fields

Table 3.54: VCPU-Scope Mutable Non-Memory State MBMD Fields Definition

Field	Offset (Bytes)	Size (Bytes)	Description	Included In MAC	Included in IV
VP_INDEX	24	2	Virtual CPU index	Yes	No
RESERVED	26	6	Reserved, must be 0	Yes	No

3.12.1.5. TD Private Memory MBMD Fields

Table 3.55: TD Private Memory MBMD Type-Specific Fields Definition

Field	Offset (Bytes)	Size (Bytes)	Description	Included In MAC	Included in IV
NUM_GPAS	24	2	Number of entries in the GPA list	Yes	No
GPA_LIST_ATTRIBUTES	26	1	Attributes of the GPA list, see Table 3.56 below	Yes	No
RESERVED	27	5	Reserved, must be 0	Yes	No

Table 3.56: GPA_LIST_ATTRIBUTES

Bits	Name	Description		
2:0	**FORMAT**	GPA list format		
		Value	**Name**	**Description**
		0	GPA_ONLY	A GPA list page is provided
		1	GPA_AND_ATTR	A GPA list page and a page attributes list page are provided
		Other	RESERVED	Reserved
7:3	**RESERVED**	Reserved: must be 0		

3.12.1.6. Epoch Token MBMD Fields

Table 3.57: Epoch Token MBMD Fields Definition

Field	Offset (Bytes)	Size (Bytes)	Description	Included In MAC	Included in IV
TOTAL_MB	24	8	The total number of migration bundles, including the current one, which have been exported since the beginning of the migration session	Yes	No

3.12.1.7. Abort Token MBMD Fields

Table 3.58: Abort Token MBMD Fields Definition

Field	Offset (Bytes)	Size (Bytes)	Description	Included In MAC	Included in IV
RESERVED	24	8	Reserved, must be 0	Yes	No

3.12.1.8. TD Migration Protocol Version Compatibility

The table below specifies the TD migration protocol versions for which the above MBMD definition is appliable.

Table 3.59: MBMD Compatibility with TD Migration Versions

TD Migration Version	Minimum	Maximum
Export version	0	0
Import version	0	0

3.12.2. GPA List

A GPA list specifies a list of GPAs to migrated by TDH.EXPORT.MEM and TDH.IMPORT.MEM, blocked for writing by TDH.EXPORT.BLOCKW or reset to their original SEPT entry state by TDH.EXPORT.RESTORE. GPA list may have up to 512 entries, is contained in a single 4KB page and must be aligned on 4KB. The GPA list may contain null entries, as indicated by OPERATION field's value set to 0 (NOP).

3.12.2.1. GPA_LIST_INFO: HPA, First and Last Entries of a GPA List

GPA_LIST_INFO is a 64b structure used as a GPR input and output operand of multiple migration interface functions, e.g., TDH.EXPORT.MEM. It provides the HPA of the GPA list page in shared memory, and the index of the first entry and last entries to be processed.

Table 3.60: GPA_LIST_INFO

Bits	Name	Description		
2:0	**FORMAT**	GPA list format		
		Value	**Name**	**Description**
		0	GPA_ONLY	A GPA list page is provided
		1	GPA_AND_L2_ATTR	GPA list and L2 page attributes list pages are provided. This format is only used by TDH.EXPORT.MEM and TDH.IMPORT.MEM. It is mandatory for migrating partitioned TDs (which contain one or more L2 VMs). TDX module support of this feature is enumerated by TDX_FEATURES0.PARTITIONED_TD_MIGRATION, readable by TDH.SYS.RD* (see 3.3.3.1).
		Other	RESERVED	Reserved
11:3	**FIRST_ENTRY**	Index of the first entry of the list to be processed		
51:12	**HPA**	Bits 51:12 of the host physical address (including HKID) of the GPA list page, which must be a shared HPA		
54:52	**RESERVED**	Reserved: must be 0		

Bits	Name	Description
63:55	**LAST_ENTRY**	Index of the last entry in the GPA list

3.12.2.2. GPA List Entry

Table 3.61 below shows the format of a GPA list entry as used. The GPA list entry format is designed so that the output of TDH.EXPORT.BLOCKW can be used directly with TDH.EXPORT.MEM, and the output of TDH.EXPORT.MEM can be used directly with TDH.IMPORT.MEM.

Table 3.61: GPA List Entry Definition

Bit(s)	Size	Name	Description	TDH.EXPORT.BLOCKW		TDH.EXPORT.MEM		TDH.IMPORT.MEM		TDH.EXPORT.RESTORE	
				In	Out	In	Out	In	Out	In	Out
1:0	2	LEVEL	Mapping level (size)	Must be 0 (4KB)	Unmod.	Must be 0 (4KB)	Unmod.	Must be 0 (4KB)	Unmod.	Must be 0 (4KB)	Unmod.
2	1	PENDING	See below	Ignored	Unmod.	Ignored	Yes	Yes	Unmod.	Ignored	Unmod.
6:3	4	RESERVED	Reserved	Must be 0	Unmod.	Must be 0	Unmod.	Must be 0	Unmod.	Must be 0	Unmod.
9:7	3	L2_MAP	See below	Ignored	Unmod.	Ignored	Yes	Yes	Unmod.	Ignore	Unmod.
11:10	2	MIG_TYPE	See below	Yes	Unmod.	Yes	Unmod.	Yes	Unmod.	Yes	Unmod.
51:12	40	GPA	Guest Physical Address bits 51:12	Yes	Unmod.	Yes	Unmod.	Yes	Unmod.	Yes	Unmod.
53:52	2	OPERATION	See below	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
55:54	2	RESERVED	Reserved	Must be 0	Unmod.	Must be 0	Unmod.	Must be 0	Unmod.	Must be 0	Unmod.
60:56	5	STATUS	See below	Ignored	Yes	Ignored	Yes	Ignored	Yes	Ignored	Yes
63:61	3	RESERVED	Reserved	Must be 0	Unmod.	Must be 0	Unmod.	Must be 0	Unmod.	Must be 0	Unmod.

3.12.2.3. GPA List Entry Details

GPA List Details: LEVEL

Reserved for future support of page sizes other than 4KB.

GPA List Details: PENDING

Table 3.62: PENDING Values Definition

Value	Name	Description
0	MAPPED	SEPT entry is MAPPED
1	PENDING	SEPT entry is PENDING

GPA List Details: L2_MAP

A bitmap with indicates whether the page is mapped in one or more L2 VM. This field is provided as part of the GPA list entry to enable the host VMM to prepare L2 SEPT pages before invoking TDH.IMPORT.MEM.

GPA List Details: OPERATION

The following tables describe the meaning of OPERATION, as used for each applicable interface function. Note that the OPERATION definitions for TDH.EXPORT.BLOCKW, TDH.EXPORT.MEM and TDH.IMPORT.MEM are designed to be compatible, so that the same GPA list can be used for all of them.

Table 3.63: OPERATION Values Definition for TDH.EXPORT.BLOCKW

Value	Input		Output	
	Name	**Description**	**Name**	**Description**
0	NOP	No operation	NOP	Not blocked for writing
1	BLOCKW	Block for writing	BLOCKW	Blocked for writing
2	NOP	No operation	NOP	Not blocked for writing
3	BLOCKW	Block for writing	BLOCKW	Blocked for writing

Table 3.64: OPERATION Values Definition for TDH.EXPORT.MEM

Value	Input		Output	
	Name	**Description**	**Name**	**Description**
0	NOP	No operation	NOP	Not exported
1	MIGRATE	Export	MIGRATE	Initial export during this migration session or following a CANCEL
2	CANCEL	Cancel previous export	CANCEL	Cancellation of a previous export Not applicable for S4 hibernation.
3	MIGRATE	Export	REMIGRATE	Re-export of updated content or attributes

Table 3.65: OPERATION Values Definition for TDH.IMPORT.MEM

Value	Input		Output	
	Name	**Description**	**Name**	**Description**
0	NOP	No operation	NOP	Not imported
1	MIGRATE	Initial import during this migration session or following a CANCEL	MIGRATE	Imported
2	CANCEL	Cancel previous import	CANCEL	Removed previous import Not applicable for S4 resumption.
3	REMIGRATE	Re-import of updated page content or attributes	REMIGRATE	Imported Not applicable for S4 resumption.

Table 3.66: OPERATION Values Definition for TDH.EXPORT.RESTORE

Value	Input		Output	
	Name	**Description**	**Name**	**Description**
0	NOP	No operation	NOP	Not restored
1	RESTORE	Restore SEPT entry to non-migration state	RESTORE	Restored
2	NOP	Reserved	NOP	Not restored
3	RESTORE	Restore SEPT entry to non-migration state	RESTORE	Restored

GPA List Details: MIG_TYPE

Table 3.67: MIG_TYPE Values Definition

Value	Name	Description
0	PAGE_4K	4KB private memory page
Other	RESERVED	Reserved for future types

GPA List Details: STATUS

Table 3.68: STATUS Values Definition

Value	Name	Description
0	SUCCESS	GPA list entry was processed successfully
1	SKIPPED	GPA list entry was skipped because NOP was requested
2	SEPT_WALK_FAILED	Secure EPT walk failed for the requested GPA
3	SEPT_ENTRY_BUSY_HOST_PRIORITY	Secure EPT entry was busy. The host VMM should retry the operation until successful.
4	SEPT_ENTRY_STATE_INCORRECT	Secure EPT entry state was incorrect for the requested operation and the TD's OP_STATE
5	TLB_TRACKING_NOT_DONE	TLB tracking was not done for the requested GPA
6	OP_STATE_INCORRECT	The TD's OP_STATE was incorrect for the requested operation and Secure EPT entry state
7	MIGRATED_IN_CURRENT_EPOCH	Requested GPA has already been migrated during the current migration epoch
8	MIG_BUFFER_NOT_AVAILABLE	Required migration buffer was not provided
9	NEW_PAGE_NOT_AVAILABLE	Required new TD page was not provided
10	INVALID_PAGE_MAC	Page MAC was invalid
11	DISALLOWED_IMPORT_OVER_REMOVED	Page import over a removed page is not allowed
12	TD_PAGE_BUSY_HOST_PRIORITY	TD page was busy. The host VMM should retry the operation until successful.
13	L2_SEPT_WALK_FAILED	L2 Secure EPT walk failed for the requested GPA
14	ATTR_LIST_ENTRY_INVALID	The L2 attributes list entry is invalid
15	GPA_LIST_ENTRY_INVALID	The GPA list entry is invalid
31-16	Reserved	Reserved

3.12.2.4. TD Migration Protocol Version Compatibility

The table below specifies the TD migration protocol versions for which the above GPA List definition is appliable.

Table 3.69: GPA List Compatibility with TD Migration Versions

TD Migration Version	Minimum	Maximum
Export version	0	0
Import version	0	0

3.12.3. Memory Migration Buffers List

A memory migration buffer list specifies a list of HPAs of 4KB pages in shared memory, to be used as output by TDH.EXPORT.MEM and as input by TDH.IMPORT.MEM. The list may have up to 512 64-bit entries, each containing a 4KB-aligned HPA (including HKID bits) of a page in shared memory. The list is contained in a single 4KB page and must be aligned on 4KB. The page list may contain null entries, indicated by the INVALID bit.

3.12.3.1. Migration Buffers List Entry

Table 3.70: Migration Buffers List Entry

Bits	Name	Description
11:0	**RESERVED**	Reserved: must be 0 (unless bit 63 indicates an invalid entry)
51:12	**HPA**	Bits 51:12 of the host physical address (including HKID) of the migration buffer page, which must be a shared HPA
62:52	**RESERVED**	Reserved: must be 0 (unless bit 63 indicates an invalid entry)
63	**INVALID**	A value of 1 indicates that this entry is invalid

3.12.4. Page Attributes List

A page attributes list specifies a list of page aliases, to be used as output by TDH.EXPORT.MEM and as input by TDH.IMPORT.MEM. The list must contain an entry for each respective GPA list entry used with the same interface functions. The list may have up to 512 64-bit entries, in page L2 attributes format as defined in 3.6.3. The list is contained in a single 4KB page and must be aligned on 4KB. A page attributes list is mandatory for migrating partitioned TDs (which contain one or more L2 VMs).

Enumeration: TDX module support of page attributes list is enumerated by TDX_FEATURES0.PARTITIONED_TD_MIGRATION, readable by TDH.SYS.RD* (see 3.3.3.1).

3.12.5. Memory Migration Page MAC List

A page MAC list specifies a list of MACs over 4KB migrated pages, their GPA list entries and (if applicable) page L2 attributes list entries, to be used as output by TDH.EXPORT.MEM and as input by TDH.IMPORT.MEM. The list must contain an entry for each respective GPA list entry used with the same interface functions. The list may have up to 256 128-bit entries, each containing a single AES-GMAC-256 of a migrated page. The list is contained in a single 4KB page and must be aligned on 4KB.

3.12.6. Non-Memory State Migration Buffers List

A non-memory state migration buffer list specifies a list of HPAs of 4KB pages in shared memory, to be used as output by TDH.EXPORT.STATE.* and as input by TDH.IMPORT.STATE.*. The list may have up to 512 64-bit entries, each containing a 4KB-aligned HPA (including HKID bits) of a page in shared memory. The list is contained in a single 4KB page and must be aligned on 4KB.

3.12.6.1. PAGE_LIST_INFO: HPA and Attributes of a Page List

PAGE_LIST_INFO is a 64b structure used as a GPR input and output operand of multiple migration interface functions, e.g., TDH.EXPORT.STATE.TD. It provides the HPA of the migration buffers list page in shared memory, and the index of the last entry to be processed.

Table 3.71: PAGE_LIST_INFO

Bits	Name	Description
11:0	**RESERVED**	Reserved: must be 0
51:12	**HPA**	Bits 51:12 of the host physical address (including HKID) of the page list, which must be a shared HPA

Bits	Name	Description
54:52	**RESERVED**	Reserved: must be 0
63:55	**LAST_ENTRY**	Index of the last entry in the page list

4. TD Metadata (Non-Memory State)

This chapter describes the details of TD metadata, a.k.a. non-memory state or control state.

4.1. TD-Scope Metadata

TD-scope control structures TDR and TDCS are described the [TDX Module Base Spec].

Information about TDR and TDCS is provided in a separate JSON format file **td_scope_metadata.json**.

4.1.1. TDR

Note: This section describes TDR, as defined. Implementation may differ.

TDR is the root control structure of a guest TD. TDR is encrypted using the Intel TDX global private HKID. It contains the minimal set of fields that allow TD management operation when the guest TD's private ephemeral HKID is not known yet or when the TD's key state is such that memory encrypted with the guest TD's private ephemeral key is not accessible.

TDR occupies a single 4KB naturally aligned page of memory. It is the first TD page to be allocated and the last to be removed. None of the state in the TDR is migrated – it is locally initialized on the destination platform for a migrated TD.

TRD fields are divided into the following classes:

Table 4.1: TDR Field Classes Definition

Field Class	Description
TD Management	These fields are used to manage the TDR page, its descendent TD private memory pages and control structure pages.
Key Management	These fields are used by the Intel TDX module to manage memory encryption keys. See the [TDX Module Base Spec] for details.
TD Preserving	These fields are used by the Intel TDX module to manage the TD across TD preserving updates.

4.1.2. TDCS

Note: This section describes TDCS, as defined. Implementation may differ.

TDCS complements TDR as the logical control structure of a guest TD. TDCS is encrypted with the guest TS's ephemeral private key. It controls the guest TD operation and holds the state that is global to all the TD's VCPUs. TDCS state fields are initialized either via TDH.MNG.INIT, or via TDH.IMPORT.STATE.IMMUTABLE – the latter when the TD is the target for migration.

TDCS fields are divided into the following classes:

Table 4.2: TDCS Field Classes Definition

Field Class	Description
TD Management	These fields are used to manage the TDCS, its descendent TD private memory pages and control structure pages.
TD Execution Control	Control the execution of the guest TD: some TD execution control fields are provided as an input to TDH.MNG.INIT, and some of those are included in the TDG.MR.REPORT.
TLB Epoch Tracking	Track the TLB epoch of the guest TD – see the [TDX Module Base Spec] for details
Measurement	TD measurement registers and associated fields – see the [TDX Module Base Spec] for details
Migration	TDCS fields that control TD migration
MIGSC Links	Links to Migration Stream Context pages

Field Class	Description
Service TD	TDCS fields that control Service TD binding and operation
MSR Bitmaps	MSR bitmaps that control VM exit from the guest TD on RDMSR/WRMSR are common to all TD VCPUs and thus are stored as part of TDCS.
Secure EPT Root Page	The root page (PML5 or PML4) of the secure EPT
L2 Secure EPT Root[3:1]	The root pages (PML5 or PML4) of the secure EPTs associated with L2 VM 1, 2 and 3

Following is some information about specific VMCS fields that is too extensive to provide in the JSON format files.

4.1.2.1. TDCS.TD_CTLS

TD_CTLS is a bitmap of TD controls that may be modified during TD run time.

Table 4.3: TDCS.TD_CTLS Definition

Bits	Name	Description
0	**PENDING_VE_DISABLE**	Controls the way guest TD access to a PENDING page is processed: **0 (default):** An EPT violation due to guest TD access to a PENDING page results in a #VE(PENDING). **1:** An EPT violation due to guest TD access to a PENDING page results in a TD exit. The above applies only to L1. L2 VM access to a PENDING pages always results in an L2→L1 exit. PENDING_VE_DISABLE's initial value is copied from the TD's ATTRIBUTES.SEPT_VE_DISABLE. If the TD's CONFIG_FLAGS.FLEXIBLE_PENDING_VE is 1, the TD is allowed to modify PENDING_VE_DISABLE using TDG.VM.WR. **Enumeration:** TDX module support of PENDING_VE_DISABLE is enumerated by TDX_FEATURES0.PENDING_EPT_VIOLATION_V2 (bit 16). If not supported, must be 0.
1	**ENUM_TOPOLOGY**	Controls the enumeration of virtual platform topology: **0 (default):** Guest TD execution of CPUID(0xB) or CPUID(0x1F) results in a #VE(CONFIG_PARAVIRT). CPUID(1).EBX[31:24] returns the least significant 8 bits of the VCPU index. RDMSR of IA32_X2APIC_APICID (0x802) results in #VE(CONFIG_PARAVIRT). **1:** Guest TD execution of CPUID(0xB) or CPUID(0x1F) returns the virtual topology information configured by the host VMM. CPUID(1).EBX[31:24] returns the least significant 8 bits of the virtual x2APIC_ID configured by the host VMM. RDMSR of IA32_X2APIC_APICID (0x802) returns the virtual x2APIC_ID. ENUM_TOPOLOGY can only be set to 1 if x2APIC_ID has been properly configured with unique values for each VCPU. The guest TD can read TDCS.TOPOLOGY_ENUM_CONFIGURED using TDG.VM.RD to check that. ENUM_TOPOLOGY is implicitly set by the TDX module if the guest TD sets REDUCE_VE. **Enumeration:** TDX module support of ENUM_TOPOLOGY is enumerated by TDX_FEATURES0.TOPOLOGY_ENUM (bit 20). If not supported, must be 0.

Bits	Name	Description
2	**VIRT_CPUID2**	Controls the virtualization of CPUID(2): **0 (default):** Guest TD execution of CPUID(2) results in a #VE(CONFIG_PARAVIRT). **1:** Guest TD execution of CPUID(2) returns fixed values EAX=0x00FEFF01, EBX=0, ECX=0 and EDX=0, meaning "cache data is returned by CPUID leaf 0x4" and "TLB data is returned by CPUID leaf 0x18". VIRT_CPUID2 is implicitly set by the TDX module if the guest TD sets REDUCE_VE. **Enumeration:** TDX module support of VIRT_CPUID2 is enumerated by TDX_FEATURES0.CPUID2_VIRT (bit 29). If not supported, must be 0.
3	**REDUCE_VE**	Allows the guest TD to control the way #VE is injected by the TDX module on guest TD execution of CPUID, RDMSR/WRMSR and other instructions: **0 (default):** No change to default behavior. **1:** #VE injected on guest TD execution of CPUID, RDMSR/WRMSR and other instructions is greatly reduced. Some #VE injection depends on CPUID configuration of paravirtualized CPU features using TDCS.FEATURE_PARAVIRT_CTRL, see 4.1.2.2 below. When the guest TD sets REDUCE_VE to 1, the TDX module also forces ENUM_TOPOLOGY and VIRT_CPUID to 1. REDUCE_VE can only be set to 1 if x2APIC_ID has been properly configured with unique values for each VCPU. The guest TD can read TDCS.TOPOLOGY_ENUM_CONFIGURED using TDG.VM.RD to check that. For a list of virtual CPUID values, MSRs and instructions impacted by REDUCE_VE, see Ch. 2. **Enumeration:** TDX module support of REDUCE_VE is enumerated by TDX_FEATURES0.VE_REDUCTION (bit 30). If not supported, must be 0.
62:4	**RESERVED**	Must be 0
63	**LOCK**	Controls locking of TD-writable virtualization controls. **0 (default):** No change to default behavior. **1:** Control fields are locked and can't be modified. The following TD-writable control fields are impacted: • TDCS.TD_CTLS • TDCS.FEATURE_PARAVIRT_CTLS • TDVPS.CPUID_SUPERVISOR_VE • TDVPS.CPUID_USER_VE • TDVPS.CPUID_CONTROL **Enumeration:** TDX module support of LOCK is enumerated by TDX_FEATURES0.VE_REDUCTION (bit 30). If not supported, must be 0.

4.1.2.2. TDCS.FEATURE_PARAVIRT_CTRL

Enumeration: Availability of FEATURE_PARAVIRT_CTRL is enumerated by TDX_FEATURES0.VE_REDUCTION (bit 30).

If TDCS.TD_CTRL.REDUCE_VE is set, the guest TD can control CPU feature paravirtualization by writing to TDCS.FEATURE_PARAVIRT_CTRL using TDG.VM.WR. The default value of TDCS.FEATURE_PARAVIRT_CTRL is all-0. The following table shows the virtualization behavior of each of the configurable CPU features, depending on the control bits combination as configured by the guest TD.

Table 4.4: TDCS.FEATURE_PARAVIRT_CTRL Definition

Bits	Paravirtualized Feature Name & Applicable Linux Kernel Feature Name	Backward Compatible (TD_CTLS.REDUCE_VE is 0)	Reduced #VE (TD_CTLS.REDUCE_VE is 1, FEATURE_PARAVIRT_CTLS bit is 0)	Reduced #VE with Paravirtualization (TD_CTLS.REDUCE_VE is 1, FEATURE_PARAVIRT_CTLS bit is 1)
0	**CORE_ CAPABILITIES** (X86_FEATURE_ CORE_ CAPABILITIES)	Controls IA32_CORE_CAPABILITIES paravirtualization, enumerated by virtual CPUID(7,0).EDX[30] (support IA32_CORE_CAPABILITIES)		
		• Virtual CPUID(7,0).EDX[30] is **forced to 1**. • Guest TD access to applicable MSRs results in a #VE(CONFIG_PARAVIRT).	• Virtual CPUID(7,0).EDX[30] is forced to 0. • Guest TD access to applicable MSRs results in a #GP(0).	• Virtual CPUID(7,0).EDX[30] is configured by the host VMM. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).
1	**DCA** (X86_FEATURE_ DCA)	Controls Direct Cache Access paravirtualization, enumerated by virtual CPUID(1).ECX[18] (DCA)		

		• Virtual CPUID(1).ECX[18] is configured by the host VMM.
		• Virtual CPUID(9) results in a #VE(CONFIG_PARAVIRT).
		• Guest TD access to applicable MSRs may result in a #GP(0) or #VE(CONFIG_PARAVIRT), depending on virtual CPUID(1).ECX[18] value.
			• Virtual CPUID(1).ECX[18] is forced to 0.
			• Virtual CPUID(9) returns all-0.
			• Guest TD access to applicable MSRs results in a #GP(0).
				• Virtual CPUID(1).ECX[18] is configured by the host VMM.
				• If configured as 0:
				  ○ Virtual CPUID(9) returns all-0.
				  ○ Guest TD access to applicable MSRs results in a #GP(0).
				• Else (configured as 1):
				  ○ Virtual CPUID(9) results in a #VE(CONFIG_PARAVIRT).
				  ○ Guest TD access to applicable MSRs results in a #VE(CONFIG_PARAVIRT).

2	**EST** (X86_FEATURE_ EST)	Controls Enhanced Intel SpeedStep technology paravirtualization, enumerated by Virtual CPUID(1).ECX[7] (Enhanced Intel SpeedStep technology)		
		• Virtual CPUID(1).ECX[7] is configured by the host VMM. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).	• Virtual CPUID(1).ECX[7] is forced to 0. • Guest TD access to applicable MSRs results in a #GP(0).	• Virtual CPUID(1).ECX[7] is configured by the host VMM. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).
3	**MCA**	Controls Machine Check Architecture paravirtualization, enumerated by virtual CPUID(1).EDX[7] (Machine Check Exception) and virtual CPUID(1).EDX[14] (Machine Check Architecture)		

Bits	Paravirtualized Feature Name & Applicable Linux Kernel Feature Name	Backward Compatible (TD_CTLS.REDUCE_VE is 0)	Reduced #VE (TD_CTLS.REDUCE_VE is 1, FEATURE_PARAVIRT_CTLS bit is 0)	Reduced #VE with Paravirtualization (TD_CTLS.REDUCE_VE is 1, FEATURE_PARAVIRT_CTLS bit is 1)
	(X86_FEATURE_ MCA)	• Virtual CPUID(1).EDX[7] and virtual CPUID(1).EDX[14] are **forced to 1**. • Guest TD access to applicable MSRs result in a #VE(CONFIG_PARAVIRT). • CR4.MCE is fixed-1. Guest TD attempt to clear it result in a #VE(CONFIG_PARAVIRT).	• Virtual CPUID(1).EDX[7] and virtual CPUID(1).EDX[14] are forced to 0. • Guest TD access to applicable MSRs results in a #GP(0). • CR4.MCE is initialized to 1. The guest TD may clear CR4.MCE but not set it back to 1; attempt to do so results in a #GP(0).	• Virtual CPUID(1).EDX[7] and virtual CPUID(1).EDX[14] are configured by the host VMM. • If virtual CPUID(1).EDX[14] is 0, guest TD access to applicable MSRs results in a #GP(0). Else, it results in a #VE(CONFIG_PARAVIRT). • CR4.MCE is initialized to 1. If virtual CPUID(1).EDX[7] is 0, the guest TD may clear CR4.MCE but not set it back to 1; attempt to do so results in a #GP(0. Else, guest TD is allowed to modify CR4.MCE.
4	**MTRR** (X86_FEATURE_ MTRR)	Controls Memory Type Range Registers paravirtualization, enumerated by virtual CPUID(1).EDX[12] (Memory Type Range Registers)		
		• Virtual CPUID(1).EDX[12] is **forced to 1**. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).	• Virtual CPUID(1).EDX[12] is forced to 0. • Guest TD access to applicable MSRs results in a #GP(0).	• Virtual CPUID(1).EDX[12] is configured by the host VMM. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).
5	**PCONFIG** (X86_FEATURE_ PCONFIG)	Controls PCONFIG paravirtualization, enumerated by virtual CPUID(7,0).EDX[18] (PCONFIG)		
		• Virtual CPUID(7,0).EDX[18] is configured by the host VMM. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).	• Virtual CPUID(7,0).EDX[18] is forced to 0. • Guest TD access to applicable MSRs results in a #GP(0).	• Virtual CPUID(7,0).EDX[18] is configured by the host VMM. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).
6	**RDT_A** (X86_FEATURE_ RDT_A)	Controls RDT-A paravirtualization, enumerated by virtual CPUID(7,0).EBX[15] (RDT-A)		
		• Virtual CPUID(7,0).EBX[15] is configured by the host VMM. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).	• Virtual CPUID(7,0).EBX[15] is forced to 0. • Guest TD access to applicable MSRs results in a #GP(0).	• Virtual CPUID(7,0).EBX[15] is configured by the host VMM. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).
7	**RDT_M** (X86_FEATURE_ CQM)	Controls RDT-M paravirtualization, enumerated by virtual CPUID(7,0).EBX[12] (RDT-M)		
		• Virtual CPUID(7,0).EBX[12] is configured by the host VMM. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).	• Virtual CPUID(7,0).EBX[12] is forced to 0. • Guest TD access to applicable MSRs results in a #GP(0).	• Virtual CPUID(7,0).EBX[12] is configured by the host VMM. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).
8	**ACPI**	Controls Thermal Monitor and Software Controlled Clock Facilities paravirtualization, enumerated by virtual CPUID(1).EDX[22] (ACPI)		

Bits	Paravirtualized Feature Name & Applicable Linux Kernel Feature Name	Backward Compatible (TD_CTLS.REDUCE_VE is 0)	Reduced #VE (TD_CTLS.REDUCE_VE is 1, FEATURE_PARAVIRT_CTLS bit is 0)	Reduced #VE with Paravirtualization (TD_CTLS.REDUCE_VE is 1, FEATURE_PARAVIRT_CTLS bit is 1)
	(X86_FEATURE_ ACPI)	• Virtual CPUID(1).EDX[22] is configured by the host VMM. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).	• Virtual CPUID(1).EDX[22] is forced to 0. • Guest TD access to applicable MSRs results in a #GP(0).	• Virtual CPUID(1).EDX[22] is configured by the host VMM. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).
9	**TM2** (X86_FEATURE_ TM2)	Controls MSR_THERM2_CTL paravirtualization, enumerated by virtual CPUID(1).ECX[8] (TM2)		
		• Virtual CPUID(1).ECX[8] is configured by the host VMM. • Guest TD access to MSR_THERM2_CTL may result in a #VE(CONFIG_PARAVIRT).	• Virtual CPUID(1).ECX[8] is forced to 0. • Guest TD access to MSR_THERM2_CTL results in a #GP(0).	• Virtual CPUID(1).ECX[8] is configured by the host VMM. • Guest TD access to MSR_THERM2_CTL may result in a #VE(CONFIG_PARAVIRT).
10	**TME** (X86_FEATURE_ TME)	Controls Total Memory Encryption paravirtualization, enumerated by virtual CPUID(7,0).ECX[13] (TME_EN)		
		• Virtual CPUID(7,0).ECX[13] is configured by the host VMM. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).	• Virtual CPUID(7,0).ECX[13] is forced to 0. • Guest TD access to applicable MSRs results in a #GP(0).	• Virtual CPUID(7,0).ECX[13] is configured by the host VMM. • Guest TD access to applicable MSRs may result in a #VE(CONFIG_PARAVIRT).
11	**TSC_DEADLINE** (X86_FEATURE_ TSC_DEADLINE_ TIMER)	Controls IA32_TSC_DEADLINE MSR paravirtualization, enumerated by virtual CPUID(1).ECX[24] (TSC deadline)		
		• Virtual CPUID(1).ECX[24] is configured by the host VMM. • Guest TD access to the IA32_TSC_DEADLINE MSR may result in a #GP(0) or #VE(CONFIG_PARAVIRT), depending on virtual CPUID(1).ECX[24].	• Virtual CPUID(1).ECX[24] is forced to 0. • Guest TD access to the IA32_TSC_DEADLINE MSR results in a #GP(0).	• Virtual CPUID(1).ECX[24] is configured by the host VMM. • Guest TD access to the IA32_TSC_DEADLINE MSR may result in a #VE(CONFIG_PARAVIRT).
63:12	**RESERVED**	Must be 0		

4.2. TDVPS: VCPU-Scope Metadata

Note: This section describes TDVPS, as defined. Implementation may differ.

TDVPS is described in the [TDX Module Base Spec].

Information about TDVPS is provided in a separate JSON format file **vcpu_scope_metadata.json**.

4.2.1. Overview

Logically, in the Intel TDX module's linear address space, TDVPS is a single structure that holds the state and control information for a single TD VCPU. The state is loaded to the LP on TD Entry and saved on TD exits.

Physically, TDVPS is composed of a root page (TDVPR) and multiple extension pages (TDCX). The pages need not be contiguous in physical memory.

TDVPS is initialized by TDH.VP.INIT. For an TD being migrated, TDVPS is imported by TDH.IMPORT.STATE.VP, which initializes some state fields and migrates some fields from the source TD VPS state.

TDVPS fields are divided into the following classes:

Table 4.5: TDVPS Field Classes Definition

Field Class	Description
VCPU Management	These fields are used to manage the TDVPS and the TD VCPU.
TD VMCS	The TD VCPU's L1 VM architectural VMCS
VAPIC	The TD VCPU's Virtual APIC page
VE_INFO	Holds Virtualization Exception (#VE) information
Guest GPR State	TD VCPU's general-purpose register state
Guest MSR State	TD VCPU's MSR state
Guest Extended State	TD VCPU's extended state
VMCS[3:1]	VMCS pages of L2 VM 1, 2 and 3
MSR Bitmaps[3:1]	MSR bitmap pages of L2 VM 1, 2 and 3
MSR Bitmaps Shadow[3:1]	MSR bitmap shadow pages of L2 VM 1, 2 and 3

4.2.2. TDVPS (excluding TD VMCS)

Information about TDVPS is provided in a separate JSON format file **vcpu_scope_metadata.json**.

4.2.3. TD (L1) VMCS and L2 VMCS

Intel SDM, Vol. 3, 24 Virtual Machine Control Structures

Note: This section describes VMCS usage, as defined. Implementation may differ.

TD (L1) VMCS and L2 VMCS are VMX-format VMCS (with TDX ISA extensions) that are stored as part of TDVPS.

Most of the information about TD VMCS and L2 VMCS is provided in separate JSON format files **td_vmcs.json** and **l2_vmcs.json**.

Following is some information about specific VMCS fields that is too extensive to provide in the JSON format files.

4.2.3.1. TD VMCS CR4 Guest/Host Mask

Table 4.6: TD VMCS CR4 Guest/Host Mask

Bit	Name	Value	Description
6	MCE	1	Owned by the Intel TDX module
13	VMXE	1	Owned by the Intel TDX module
14	SMXE	1	Owned by the Intel TDX module
19	KL	~TDCS.ATTRIBUTES.KL	Intercept writes to CR4 if KeyLocker is not enabled
22	PKE	~TDCS.XFAM[9]	Intercept writes to CR4 If PK is not enabled
24	PKS	~TDCS.ATTRIBUTES.PKS	Intercept writes to CR4 if PKS is not enabled
25	UINTR	~TDCS.XFAM[14]	Intercept writes to CR4 if ULI is not enabled
27	LASS	~TDCS.ATTRIBUTES.LASS	Intercept writes to CR4 if LASS is not enabled
32	FRED	~virt. CPUID(0x7,1).EAX[17]	Intercept writes to CR4 if FRED is not enabled Applicable to TDX modules which support FRED.

Bit	Name	Value	Description
Any bit set to 1 in IA32_VMX_CR4_FIXED0 (i.e., a bit whose value must be 1)		1	Intercept writes of illegal values to CR4
Any bit set to 0 in IA32_VMX_CR4_FIXED1 (i.e., a bit whose value must be 0)		1	Intercept writes of illegal values to CR4
Bits known to the Intel TDX module as reserved (bits 63:33, 31:29, 26 and bit 15)		1	Intercept writes of illegal values to CR4
Other bits		0	

5. Interface Functions

5.1. How to Read the Interface Function Definitions

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

A table of operands is provided for any function that has explicit and/or implicit memory operands or implicit resources. Table 5.1 below describes how to read it. Most of the background is detailed in the [TDX Module Base Spec].

Table 5.1: How to Read the Operands Information Tables

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Alignment Check	Concurrency Restrictions		
								Resource	Contain. 2MB	Contain. 1GB
The operand may be specified explicitly or may be implicit	Register used as a pointer to the operand	How the operand is referenced: HPA, GPA, GPA and level or index	Resource (memory or CPU internal) for this operand	Data type of the resource, as defined in Chapter 3 or Chapter 4	Type of memory or resource access: R, RW, or Ref	Shared, Private, Opaque or Hidden	Required alignment of the operand	Concurrency restrictions are described in the [TDX Module Base Spec]. For explicit memory accesses using HPA, there are additional concurrency restrictions on the 1GB and 2MB blocks that contain the accessed HPA. For other types of accesses, only the operand concurrency is applicable. Shared(h) and Exclusive(h) indicate shared access with host-side priority. Sh./Ex.(h) indicates either shared or exclusive access with host-side priority, depending on the platform type • On platforms which do not use ACT, access is shared. • On platforms which use ACT, access is exclusive. Shared(i) and Exclusive(i) indicate that the resource is implicitly restricted.		

5.2. Reserved Leaf Numbers

The following SEAMCALL and TDCALL leaf number ranges are reserved and will never be used by production TDX modules:

0x00F0 - 0x00FF: Range reserved for debug builds of the TDX module

0xF000 - 0xFFFF: Range reserved for debug builds of the TDX module

0xE000 - 0xEFFF: Range reserved for software use, will never be used by the TDX module

Other: Ranges available for SEAMCALL and TDCALL leaf assignments

5.3. Common Algorithms Used by Multiple Interface Functions

This section describes common algorithms that are used by multiple interface functions.

5.3.1. VCPU Association with an LP

The following algorithm is used for associating the current VCPU with the current LP. It is used with VCPU-specific host-side interface functions such as TDH.VP.ENTER, TDH.VP.RD etc.

1. Check that the VCPU has been initialized and is not being torn down.
2. Atomically check that the VCPU is not associated with another LP and associate it with the current LP.
3. If this is a new association, and the TD's ephemeral HKID has changed since last association, update all TD VMCS physical pointers and the TD HKID execution control.
4. Update the TD VMCS host state fields with any Intel TDX module LP-specific values.

5.3.2. Metadata Access

5.3.2.1. Single Metadata Field Read

The following algorithm is used when reading a single metadata field based on a provided field identifier. This algorithm is used by TDH.MNG.RD, TDH.VP.RD and TDG.VM.RD, TDG.VP.RD.

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

1. Check that the field identifier is valid and derive a read mask depending on whether this algorithm is used by a host-side or a guest-side interface function, and whether the TD runs in debug mode (ATTRIBUTES.DEBUG is 1).
2. If the read mask is 0, then fail; the field in not readable.

If the above checks passed:

3. Read the field value from the control structure using the proper method per field class.
4. Mask the field value with the read mask derived above and return the resulting value.
   4.1. In some cases, special handling is required. E.g., the field value may need to be translated to another format, or some other action may be needed.

5.3.2.2. Single Metadata Field Write

The following algorithm is used when writing a single metadata field based on provided field identifier, input value and write mask. This algorithm is used by TDH.MNG.WR, TDH.VP.WR, TDG.VM.WR, TDG.VP.WR and TDG.SERVTD.WR.

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

1. Check that the field identifier is valid and derive the field attributes (read mask, write mask) depending on whether this algorithm is used by a host-side, guest-side or service TD interface function, and whether the TD runs in debug mode (ATTRIBUTES.DEBUG is 1).
2. If the write mask is 0, then fail, the field in not writable.

If passed:

3. Calculate a combined write mask:
   3.1. If a write mask is provided as an input, derive the combined write mask by bitwise-anding the field's write mask derived above with the write mask provided as an input.
   3.2. Else (no write mask input), the combined write mask is set to the field's write mask.
4. If the combined write mask is 0, then fail, the field in not writable.

If passed:

5. Read the old field value from the control structure using the proper method per field class.
6. Check for write validity. The caller must not attempt to modify any non-writable bit that is readable.
   6.1. A bit N is considered "forbidden" if it was requested to be written (i.e., the input write mask bit N is 1) but is not writable (i.e., the field's write mask bit N is 0).
   6.2. If a bit N is forbidden, then the input bit N's value must be the same as the current field's bit N's value as would be read by the caller, i.e., taking into account the field's read mask.
7. Calculate a new field value based on the input value and the combined write mask.
   7.1. Bits for which the combined write mask's value is 1 are taken from the input value.
   7.2. Other bits are taken from the current field's value.

8. Write the field value to the control structure using the proper method per field class.
   8.1. In some cases, special handling is required. E.g., the new field value may need to be checked for validity, or some other action may be needed.

If passed:

9. Mask the old field value with the field's read mask derived above, and return the resulting value.

5.3.2.3. Multiple Metadata Fields Write based on a Metadata List

The following algorithm is used when writing multiple metadata fields based on a provided metadata list. This algorithm is used by TDH.IMPORT.STATE.*.

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

1. Check the list header to be valid (NUM_SEQUENCES > 0).

If passed:

2. For each sequence in the list:
   2.1. Check that the list did not cross 4KB page boundary.
   2.2. Read the sequence header and check it is valid.

   If the above checks passed:

   2.3. For each field in the sequence:
      2.3.1. Check that the list did not cross 4KB page boundary.
      2.3.2. Check that the field identifier is valid and derive a write mask depending on whether this algorithm is used by a host-side or a guest-side interface function, and whether the TD runs in debug mode (ATTRIBUTES.DEBUG is 1).
      2.3.3. If the write mask is 0, then fail, the field in not writable.

      If the above checks passed:

      2.3.4. Calculate an effective write mask:
         2.3.4.1. If a write mask is provided for each field in the current sequence, derive the effective write mask by bitwise-anding the write mask derived above with the write mask provided with the field.
         2.3.4.2. Else, the effective write mask is the write mask derived above.
      2.3.5. If the effective write mask is 0, then fail, the field in not writable.

      If passed:

      2.3.6. Read the existing field value from the control structure using the proper method per field class.
      2.3.7. Calculate a new field value based on the input value and the effective write mask, and write to the control structure using the proper method per field class.
         2.3.7.1. In some cases, special handling is required. E.g., the new field value may need to be checked for validity, or some other action may be needed.

5.4. Host-Side (SEAMCALL) Interface Functions

The SEAMCALL instruction enters the Intel TDX module. It is designed to call host-side Intel TDX functions, either local or a TD entry to a guest TD, as selected by RAX.

5.4.1. SEAMCALL Instruction (Common)

This section describes the common functionality of SEAMCALL. Leaf functions are described in the following sections.

Table 5.2: SEAMCALL Input Operands Definition

Parameter	Description		
RAX	Leaf and version numbers, as defined in the [TDX Module Base Spec]. See Table 5.4 below for SEAMCALL leaf numbers.		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version
	63:24	Reserved	Must be 0
Other	See individual SEAMCALL leaf functions.		

Table 5.3: SEAMCALL Output Operands Definition

Parameter	Description
RAX	If SEAMCALL failed with VMfailInvalid condition (RFLAGS.CF is 1), then RAX is unmodified. Else, if on input RAX bit 63 was 1, then the SEAMCALL leaf function has been processed by the P-SEAMLDR. Refer to the [TDX Loader Spec] for details. Else, RAX contains the leaf function status return code, indicating the outcome of execution of the SEAMCALL leaf function. See the [TDX Module Base Spec] for details
Other	See individual SEAMCALL leaf functions.

Table 5.4: SEAMCALL Instruction Leaf Numbers Definition

Leaf #	Interface Function Name	Description
0	TDH.VP.ENTER	Enter TDX non-root operation
1	TDH.MNG.ADDCX	Add a control structure page to a TD
2	TDH.MEM.PAGE.ADD	Add a 4KB private page to a TD during TD build time
3	TDH.MEM.SEPT.ADD	Add and map a 4KB Secure EPT page to a TD
4	TDH.VP.ADDCX	Add a control structure page to a TD VCPU
5	TDH.MEM.PAGE.RELOCATE	Relocate a 4KB mapped page from its HPA to another
6	TDH.MEM.PAGE.AUG	Dynamically add a 4KB private page to an initialized TD
7	TDH.MEM.RANGE.BLOCK	Block a TD private GPA range
8	TDH.MNG.KEY.CONFIG	Configure the TD private key on a single package
9	TDH.MNG.CREATE	Create a guest TD and its TDR root page
10	TDH.VP.CREATE	Create a guest TD VCPU and its TDVPR root page
11	TDH.MNG.RD	Read TD metadata
12	TDH.MEM.RD	Read from private memory of a debuggable guest TD
13	TDH.MNG.WR	Write TD metadata
14	TDH.MEM.WR	Write to private memory of a debuggable guest TD

Leaf #	Interface Function Name	Description
15	TDH.MEM.PAGE.DEMOTE	Split a 2MB or a 1GB private TD page mapping into 512 4KB or 2MB page mappings respectively
16	TDH.MR.EXTEND	Extend the guest TD measurement register during TD build
17	TDH.MR.FINALIZE	Finalize the guest TD measurement register
18	TDH.VP.FLUSH	Flush the address translation caches and cached TD VMCS associated with a TD VCPU
19	TDH.MNG.VPFLUSHDONE	Check all of a guest TD's VCPUs have been flushed by TDH.VP.FLUSH
20	TDH.MNG.KEY.FREEID	Mark the guest TD's HKID as free
21	TDH.MNG.INIT	Initialize per-TD control structures
22	TDH.VP.INIT	Initialize the per-VCPU control structures
23	TDH.MEM.PAGE.PROMOTE	Merge 512 consecutive 4KB or 2MB private TD page mappings into one 2MB or 1GB page mapping respectively
24	TDH.PHYMEM.PAGE.RDMD	Read the metadata of a page in a TDMR
25	TDH.MEM.SEPT.RD	Read a Secure EPT entry
26	TDH.VP.RD	Read VCPU metadata
27	TDH.MNG.KEY.RECLAIMID	Does nothing; provided for backward compatibility
28	TDH.PHYMEM.PAGE.RECLAIM	Reclaim a physical memory page owned by a TD (i.e., TD private page, Secure EPT page or a control structure page)
29	TDH.MEM.PAGE.REMOVE	Remove a private page from a guest TD
30	TDH.MEM.SEPT.REMOVE	Remove a Secure EPT page from a TD
31	TDH.SYS.KEY.CONFIG	Configure the Intel TDX global private key on the current package
32	TDH.SYS.INFO	Get Intel TDX module information
33	TDH.SYS.INIT	Globally initialize the Intel TDX module
34	TDH.SYS.RD	Read a TDX Module global-scope metadata field
35	TDH.SYS.LP.INIT	Initialize the Intel TDX module per logical processor
36	TDH.SYS.TDMR.INIT	Partially initialize a Trust Domain Memory Region (TDMR)
37	TDH.SYS.RDALL	Read all host-readable TDX Module global-scope metadata fields
38	TDH.MEM.TRACK	Increment the TD's TLB tracking counter
39	TDH.MEM.RANGE.UNBLOCK	Remove the blocking of a TD private GPA range
40	TDH.PHYMEM.CACHE.WB	Write back the contents of the cache on a package
41	TDH.PHYMEM.PAGE.WBINVD	Write back and invalidate all cache lines associated with the specified memory page and HKID
43	TDH.VP.WR	Write VCPU metadata
44	TDH.SYS.LP.SHUTDOWN	Does nothing; provided for backward compatibility
45	TDH.SYS.CONFIG	Globally configure the Intel TDX module
48	TDH.SERVTD.BIND	Bind a service TD to a target TD
49	TDH.SERVTD.PREBIND	Pre-bind a service TD to a target TD
52	TDH.SYS.SHUTDOWN	Shutdown the Intel TDX module and prepare handoff data
53	TDH.SYS.UPDATE	Populate Intel TDX module state from handoff data
64	TDH.EXPORT.ABORT	Abort an export session
65	TDH.EXPORT.BLOCKW	Block a TD private page for writing
66	TDH.EXPORT.RESTORE	Restore a list of TD private 4KB pages' Secure EPT entry states after an export abort
68	TDH.EXPORT.MEM	Export a list of TD private pages contents and/or cancellation requests
70	TDH.EXPORT.PAUSE	Pause the exported TD
71	TDH.EXPORT.TRACK	End the current in-order export phase epoch and either start a new epoch or start the out-of-order export phase
72	TDH.EXPORT.STATE.IMMUTABLE	Start an export session and export the TD's immutable state
73	TDH.EXPORT.STATE.TD	Export the TD's mutable state
74	TDH.EXPORT.STATE.VP	Export a VCPU mutable state
75	TDH.EXPORT.UNBLOCKW	Unblock a page that has been blocked for writing
80	TDH.IMPORT.ABORT	Abort an import session
81	TDH.IMPORT.END	End an import session

Leaf #	Interface Function Name	Description
82	TDH.IMPORT.COMMIT	Commit the import session and allow the imported TD to run
83	TDH.IMPORT.MEM	Import a list of TD private pages contents and/or cancellation requests based on a migration bundle in shared memory
84	TDH.IMPORT.TRACK	End the current in-order import phase epoch and either start a new epoch or start the out-of-order import phase
85	TDH.IMPORT.STATE.IMMUTABLE	Start an import session and import the TD's immutable state
86	TDH.IMPORT.STATE.TD	Import the TD's mutable state
87	TDH.IMPORT.STATE.VP	Import a VCPU mutable state
96	TDH.MIG.STREAM.CREATE	Create a migration stream

Instruction Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

The SEAMCALL instruction itself is specified in the [TDX Arch Extensions Spec]. There are multiple cases where SEAMCALL may fail. Failures may result in an exception (#UD, #GP(0)) or a VMfailInvalid condition (CF is set to 1). Failure cases include, among other, the following:

- CPU mode is incorrect
- TDX module has not been loaded
- TDX module has been disabled

If RAX bit 63 is 1, then the SEAMCALL leaf function is processed by the P-SEAMLDR. Refer to the [TDX Loader Spec] for details.

On entry, the Intel TDX module performs the checks listed below at a high level. Errors cause a SEAMRET with RAX set to the proper completion status code.

1. The leaf number in RAX is supported by the Intel TDX module.
2. If the Intel TDX module's state is not SYS_READY, only TDH.SYS.RD*, TDH.SYS.INFO, TDH.SYS.INIT, TDH.SYS.LP.INIT, TDH.SYS.CONFIG, TDH.SYS.KEY.CONFIG and TDH.SYS.SHUTDOWN leaf functions are allowed. Those leaf functions then perform other initialization state checks.

If all checks pass, the Intel TDX module calls the leaf function according to the leaf number in RAX. See the following sections for individual leaf function details.

Completion Status Codes

Table 5.5: SEAMCALL Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_SUCCESS	SEAMCALL is successful.
TDX_SYS_SHUTDOWN	
Other	See individual leaf functions.

5.4.2. TDH.EXPORT.ABORT Leaf

TDH.EXPORT.ABORT aborts an export session and allows the source TD to resume normal operation, depending on export state and an abort token received from the destination platform.

Table 5.6: TDH.EXPORT.ABORT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	HPA of the source TD's TDR page (HKID bits must be 0)		
R8	If an abort token is available, R8 provides the HPA and size of memory of an MBMD structure in memory, as described below. Otherwise, R8's value must be 0.		
	Bits	**Name**	**Description**
	51:0	HPA	Bits 51:0 of the host physical address (including HKID bits)
	63:52	Size	Size of the memory buffer containing MBMD, in bytes
R10	Migration stream index:		
	Bits	**Name**	**Description**
	15:0	MIGS_INDEX	Migration stream index – must be 0
	63:16	RESERVED	Reserved: must be 0

Table 5.7: TDH.EXPORT.ABORT Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

Leaf Function Description

TDH.EXPORT.ABORT aborts an export session. If successful, i.e., the target TD does not run, the source TD becomes runnable. If called during the out-of-order phase, an abort token received from the destination platform is required.

Enumeration: Availability of TDH.EXPORT.ABORT is enumerated by TDX_FEATURES0.TD_MIGRATION (bit 0), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.EXPORT.ABORT returns a TDX_OPERAND_INVALID(RAX) status.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.8: TDH.EXPORT.ABORT Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Explicit	R8	HPA	MBMD buffer	MBMD	R	Shared	128B	None	None	None
Explicit	R10	Index	Mig. Stream context	Mig. Stream context	RW	Opaque	N/A	Exclusive	N/A	N/A
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	Exclusive	N/A	N/A

TDH.EXPORT.ABORT checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS has been allocated (TDR.NUM_TDCX is the required number).
5. An export session is in progress but has not been committed yet: TDCS.OP_STATE is LIVE_EXPORT, PAUSED_EXPORT or POST_EXPORT.
6. The migration stream index is 0.

If successful, the function does the following:

7. If the export session is in the post-copy phase (TDCS.OP_STATE is POST_EXPORT):
   7.1. Check that the buffer provided for MBMD is large enough.
   7.2. Copy the MBMD into a temporary buffer.
   7.3. Check the MBMD fields.

   If passed:

   7.4. If the migration stream has not been initialized, initialize it.
   7.5. Build the 96b IV for this migration bundle by concatenating the stream index and the MBMD's IV_COUNTER.
   7.6. Calculate MAC based on the MAC'ed fields of MBMD and check that its value is the same as the MBMD's MAC field's value.
8. Else (the export session is in the pre-copy phase – TDCS.OP_STATE is LIVE_EXPORT or PAUSED_EXPORT):
   8.1. Check that the MBMD HPA and size provided in R8 is 0.
   8.2. Check that the migration stream index provided in R10 is 0.

If passed:

9. Terminate the export session:
   9.1. Set all migration streams' INITIALIZED and ENABLED flags to FALSE.
   9.2. Set TDCS.OP_STATE to RUNNABLE.

Completion Status Codes

Table 5.9: TDH.EXPORT.ABORT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INCORRECT_MBMD_MAC	
TDX_INVALID_MBMD	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.3. TDH.EXPORT.BLOCKW Leaf

Block a list of TD private 4KB pages for writing and for attributes modification.

Table 5.10: TDH.EXPORT.BLOCKW Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	SEAMCALL instruction leaf number and version, see 5.4.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the SEAMCALL interface function
		23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
		63:24	Reserved	Must be 0
RCX	GPA_LIST_INFO	GPA_LIST_INFO: HPA of a GPA list page in shared memory, and first and last entries to process, as defined in 3.12.2 FORMAT must be GPA_ONLY.		
RDX	TDR	HPA of the source TD's TDR page (HKID bits must be 0)		

Table 5.11: TDH.EXPORT.BLOCKW Output Operands Definition

Operand	Name	Description
RAX	STATUS	SEAMCALL instruction return code, see 5.4.1
RCX	GPA_LIST_INFO	Same as the input value, except that FIRST_ENTRY is updated to the index of the next entry to be processed. If all entries have been processed, FIRST_ENTRY is updated to (LAST_ENTRY + 1) Modulo 512.
Other		Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

For each 4KB page in the GPA list, if a blocking operation has been requested, TDH.EXPORT.BLOCKW finds the Secure EPT entry for the provided page. If the entry state is correct (MAPPED, PENDING, EXPORTED_DIRTY or PENDING_EXPORTED_DIRTY), TDH.EXPORT.BLOCKW blocks it for writing, by saving and clearing the W bit and setting the Secure EPT entry state (to BLOCKEDW, PENDING_BLOCKEDW, EXPORTED_DIRTY_BLOCKEDW or PENDING_EXPORTED_DIRTY_BLOCKEDW respectively). It records the current TD's TLB epoch in the TD's global BW_EPOCH and marks the GPA list entry as ready for export. If the TD is partitioned, TDH.EXPORT.BLOCKW also blocks any L2 SEPT entries mapping the 4KB page.

Enumeration: Availability of TDH.EXPORT.BLOCKW is enumerated by TDX_FEATURES0.TD_MIGRATION (bit 0), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.EXPORT.BLOCKW returns a TDX_OPERAND_INVALID(RAX) status.

List Entry Error: If a page can't be blocked for writing, TDH.EXPORT.BLOCKW marks its GPA list entry as unsuccessful. List processing is not aborted, it continues to the next entry, if applicable. The return status in RAX indicates the number of such cases encountered during operation.

Interruptibility: TDH.EXPORT.BLOCKW is interruptible. If a pending interrupt is detected during operation, TDH.EXPORT.BLOCKW returns with a TDX_INTERRUPTED_RESUMABLE status in RAX. RCX is updated

with the next list entry index to process, so the host BMM may re-invoke TDH.EXPORT.BLOCKW immediately after handling the interrupt.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.12: TDH.EXPORT.BLOCKW Memory Operands Information

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	GPA List page	GPA_LIST	RW	Shared	4KB	None	None	None
Explicit	RDX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Explicit	N/A	GPA	TD private pages (via GPA list)	Block	None	Private	4KB	None	None	None
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	R	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	None	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive(h)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT trees	N/A	RW	Private	N/A	Shared(i)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A

TDH.MEM.RANGE.BLOCKW checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. Export session is in the in-order phase and the TD has not been paused yet (TDCS.OP_STATE is LIVE_EXPORT).

If passed, process the GPA list:

Note: Error conditions that impact a single GPA list entry do not cause an abort of TDH.EXPORT.BLOCKW. Instead, the GPA list entry is updates with a proper status code, and the corresponding migration buffer list entry is marked as invalid.

6. For each entry in the GPA list, starting with RCX.FIRST_ENTRY and ending with RCX.LAST_ENTRY, if OPERATION indicates a BLOCKW request:
   6.1. Check the GPA list entry fields value.

   If passed:

   6.2. Walk the L1 Secure EPT based on the GPA operand and find the Secure EPT entry to be blocked.
   6.3. Check the Secure EPT entry state: it should be either of MAPPED, PENDING, EXPORTED_DIRTY or PENDING_EXPORTED_DIRTY.

6.4. If passed, update the SEPT entry and record the TD epoch:
  6.4.1. Save the original value of SEPT.W into SEPT.TDW.
  6.4.2. Clear SEPT.W.
  6.4.3. Atomically set the SEPT entry state to BLOCKEDW, PENDING_BLOCKEDW, EXPORTED_DIRTY_BLOCKEDW or PENDING_EXPORTED_DIRTY_BLOCKEDW as appropriate.
  6.4.4. If the page state is MAPPED or EXPORTED_DIRTY, then for each L2 mapping of the page:
    6.4.4.1. Walk the L2 SEPT tree based on the GPA operand and find the L2 Secure EPT entry to be blocked for writing.
    6.4.4.2. Save the original value of SEPT.W into SEPT.TDW.
    6.4.4.3. Clear SEPT.W.
    6.4.4.4. Set the L2 SEPT entry state to L2_BLOCKED

    Note: If the page state is one of the PENDING* states, then the L2 SEPT entry state is already L2_BLOCKED, no change is required.

  6.4.5. Copy the TD's epoch (TDCS.TD_EPOCH) to TDCS.BW_EPOCH.
6.5. Else:
  6.5.1. Set the GPA list entry's OPERATION field to NOP and STATUS field to the applicable status.
6.6. If this is not the last entry in the list, and there is a pending interrupt, terminate TDH.EXPORT.BLOCKW with a TDX_INTERRUPTED_RESUMABLE status.

Completion Status Codes

Table 5.13: TDH.EXPORT.BLOCKW Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_STATE_INCORRECT	
TDX_EPT_WALK_FAILED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	Operation is successful. **Note:** Processing of some GPA list entries may have encountered errors, but this did not cause an abort of the overall operation. The number such errors is reported in the lower 32 bits of the completion status.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.4. TDH.EXPORT.MEM Leaf

TDH.EXPORT.MEM exports a list of TD private pages contents and/or cancellation requests and prepares a migration bundle in shared memory.

Table 5.14: TDH.EXPORT.MEM Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	SEAMCALL instruction leaf number and version, see 5.4.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the SEAMCALL interface function
		23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
		63:24	Reserved	Must be 0
RCX	GPA_LIST_INFO	HPA of a GPA list page in shared memory, and first and last entries to process, as defined in 3.12.2 On a new invocation, FIRST_ENTRY must be 0. On a resumed invocation, FIRST_ENTRY must be the index of the next GPA list entry to export.		
RDX	TDR	HPA of the source TD's TDR page (HKID bits must be 0)		
R8	MBMD	HPA and size of memory of a memory buffer to use for MBMD:		
		Bits	**Name**	**Description**
		51:0	HPA	Bits 51:0 of the host physical address (including HKID bits)
		63:52	Size	Size of the memory buffer containing MBMD, in bytes
R9	MIG_BUFF_LIST	HPA (including HKID bits) of a migration buffer list in shared memory, corresponding to the GPA list pointed by RCX – see 3.12.3.		
R10	MIG_STREAM	Migration stream and resume flag:		
		Bits	**Name**	**Description**
		15:0	MIGS_INDEX	Migration stream index If N migration streams have been created by TDH.MIG.STREAM.CREATE, then MIGS_INDEX must be lower than N-1.
		62:16	RESERVED	Reserved: must be 0
		63	RESUME	0: This is a new invocation 1: This is resumption of a previously interrupted operation
R11	MAC_LIST_0	HPA (including HKID bits) of a MAC list in shared memory, corresponding to the first 256 entries of the GPA list pointed by RCX – see 3.12.3. If GPA_LIST_INFO.FIRST_ENTRY >= 256, then MAC_LIST_0 is ignored.		
R12	MAC_LIST_1	HPA (including HKID bits) of a MAC list in shared memory, corresponding to the last 256 entries of the GPA list pointed by RCX – see 3.12.3. If GPA_LIST_INFO.LAST_ENTRY < 256, then MAC_LIST_1 is ignored.		

Operand	Name	Description
R14	ATTRIB_LIST	If GPA_LIST_INFO.FORMAT is GPA_AND_L2_ATTR, then R14 contains the HPA (including HKID bits) of a page attributes list in shared memory – see 3.12.4. Else, R14 is ignored. An ATTRIB_LIST is mandatory for exporting partitioned TDs (which contain one or more L2 VMs). **Enumeration:** Support of this feature is enumerated by TDX_FEATURES0.PARTITIONED_TD_MIGRATION, readable by TDH.SYS.RD* (see 3.3.3.1).

Table 5.15: TDH.EXPORT.MEM Output Operands Definition

Operand	Name	Description
RAX	STATUS	SEAMCALL instruction return code, see 5.4.1
RCX	GPA_LIST_INFO	Same as the input value, except that FIRST_ENTRY is updated to the index of the next entry to be processed. If all entries have been processed, FIRST_ENTRY is updated to (LAST_ENTRY + 1) Modulo 512.
RDX	NUM_EXPORTED	If TDH.EXPORT.MEM is successful, RDX returns the number of exported 4KB migration buffers, including: • The GPA list page • One or two MAC pages (depending on GPA_LIST_INFO.FIRST_ENTRY and GPA_LIST_INFO.LAST_ENTRY • Up to 512 encrypted memory pages If TDH.EXPORT.MEM is not successful, RDX is unmodified.
AVX, AVX2 and AVX512 state		May be reset to the architectural RESET state
Other		Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

Enumeration: Availability of TDH.EXPORT.MEM is enumerated by either TDX_FEATURES0.TD_MIGRATION (bit 0) or TDX_FEATURES0.S4 (bit 13), readable by TDH.SYS.RD* (see 3.3.3.1), being set to 1. If not supported, calling TDH.EXPORT.MEM returns a TDX_OPERAND_INVALID(RAX) status.

TDX_FEATURES0.PARTITIONED_TD_MIGRATION (bit 21) enumerates TDX module support of migrating partitioned TDs (which contain one or more L2 VMs).

TDH.EXPORT.MEM exports a list of up to 512 TD private 4KB pages as a migration bundle, which includes an MBMD, set of 4KB pages encrypted with the migration session key, a 4KB page containing the GPA list, an optional 4KB containing page attributes list, and two 4KB pages containing page MACs.

A GPA list is provided as an input. For each page in the list, the requested operation may be either of the following:

- Export the page (applies also to re-exporting a previously exported page).
- Cancel a previous page export.

It is also possible to skip entries in the list by requesting no operation for specific entries. The GPA list format is described in 3.12.2. It is designed to be compatible with the output of TDH.EXPORT.BLOCKW and the input of TDH.EXPORT.RESTORE.

A list of 4KB page buffers is provided as an input. In case no data is exported (PENDING page, page cancellation or some state error) TDH.EXPORT.PAGE marks the applicable list entry as invalid.

Blocking and TLB Tracking: If the TD may be running, the exported pages must be blocked and TLB tracked. Else (e.g., the TD has been paused for export), no blocking and tracking is required.

S4 Hibernation: If TDH.EXPORT.MEM is called as part of an S4 hibernation, it only supports the out-of-order export phase. As a result, the GPA list may not contain a CANCEL operation. In addition, blocking and TLB tracking is not required.

Export Error: If a page can't be exported, TDH.EXPORT.MEM marks its GPA list entry as unsuccessful, but does not abort. It continues to the next entry, if applicable. The return status in RAX indicates the number of such cases encountered during operation.

Interruptibility: TDH.EXPORT.MEM is interruptible. If a pending interrupt is detected during operation, TDH.EXPORT.MEM returns with a TDX_INTERRUPTED_RESUMABLE status in RAX. RCX is updated with the next list entry index to process, so the host VMM may re-invoke TDH.EXPORT.MEM immediately after handling the interrupt, keeping the same inputs except setting R10.RESUME to 1.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.16: TDH.EXPORT.MEM Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	GPA List page	GPA_LIST	RW	Shared	4KB	None	None	None
Explicit	RDX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Explicit	R8	HPA	Memory to use for MBMD	MBMD	RW	Shared	128B	None	None	None
Explicit	R9	HPA	Migration buffer list	PAGE_LIST	RW	Shared	4KB	None	None	None
Explicit	R10	Index	Mig. Stream context	Mig. Stream context	RW	Opaque	N/A	Exclusive	N/A	N/A
Explicit	R11	HPA	MAC list page 1	MAC list	RW	Shared	4KB	None	None	None
Explicit	R12	HPA	MAC list page 2	MAC list	RW	Shared	4KB	None	None	None
Explicit	R14	HPA	attributes list page	page attributes	R	Shared	4KB	None	None	None
Explicit	N/A	GPA	TD private pages (via GPA list)	Blob	R	Private	4KB	None	None	None
Explicit	N/A	HPA	Migration buffer pages (via page list)	Blob	RW	Shared	4KB	None	None	None
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	R	Opaque	N/A	Shared	N/A	N/A

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive(h)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT trees	N/A	RW	Private	N/A	Shared(i)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A

TDH.EXPORT.MEM checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. An export session is in progress.
6. The migration stream index is lower than TDCS.NUM_MIGS.
7. The buffer provided for MBMD is large enough.

If successful, the function does the following:

8. If the RESUME input flag is 0, indicating that this is a new (not resumed) invocation of TDH.EXPORT.MEM:
   - 8.1. If the migration stream has not been initialized, initialize it.
   - 8.2. Increment the migration stream context's IV_COUNTER
   - 8.3. Build the 96b IV for this migration bundle by concatenating the stream index and the stream context's IV_COUNTER.
   - 8.4. Build a local copy of the MBMD.
   - 8.5. Calculate the MBMD MAC.
   - 8.6. Write the MBMD to memory.
9. Else (this is a resumption of a previously interrupted TDH.EXPORT.MEM):
   - 9.1. Check that the migration stream has been initialized.
   - 9.2. Check that the stream context's INTERRUPTED_FUNC contains TDH.EXPORT.MEM's leaf number.
   - 9.3. Check that the current inputs are the same as saved in the stream context when the function was interrupted.

If passed, process the GPA list:

Note: Error conditions that impact a single GPA list entry do not cause an abort of TDH.EXPORT.MEM. Instead, the GPA list entry is updates with a proper status code, and the corresponding migration buffer list entry is marked as invalid.

10. For each entry in the GPA list, starting with RCX.FIRST_ENTRY and ending with RCX.LAST_ENTRY:
    - 10.1. If no operation is requested, mark the corresponding migration buffer list entry as invalid and continue to the next GPA list entry.
    - 10.2. Check the GPA list entry fields value.

    If passed:

    - 10.3. Walk the L1 SEPT tree based on the GPA and level operands and find the leaf entry for the page.
    - 10.4. Check that the SEPT entry state is allowed for page export.

10.5. If the TD is running (TDCS.OP_STATE is LIVE_EXPORT) and TLB tracking is required, check TLB tracking vs. TDCS.BW_EPOCH set previously by TDH.EXPORT.BLOCKW.
10.6. Check that the requested operation is allowed in the current export phase.
10.7. Check that the requested operation is allowed for the current SEPT entry state.
10.8. If the page has any L2 mappings, check that a page attributes list has been provided (GPA_LIST_INFO.FORMAT is GPA_AND_ATTR).

Note: TDH.EXPORT.MEM does not check that the page has not been already exported in the current migration epoch during the in-order phase. This is checked when the page is imported by TDH.IMPORT.MEM.

10.9. Initialize the page attributes list entry to 0.
10.10. If passed:
10.10.1. For each L2 mapping of the page:
10.10.1.1. Walk the L2 SEPT based on the GPA and level operands and find the leaf entry for the page (if any).
10.10.2. Update the page attributes list entry.
10.10.3. Update the L1 SEPT entry state, GPA list entry and migration buffer list entry.
10.11. Else:
10.11.1. Update the GPA list entry and migration buffer list entry with error status.
10.12. Increment the migration stream context's IV_COUNTER
10.13. Build the 96b IV for this page by concatenating 0 as the direction bit, the stream index and the stream context's IV_COUNTER.
10.14. Accumulate page MAC based on the GPA list entry.
10.15. If a page attributes list has been provided, accumulate MAC based on the page attributes list entry.
10.16. If the page content is to be exported, encrypt the TD private page into the migration buffer and accumulate MAC.
10.17. Write the page MAC to the MAC list.
10.18. If this is not the last round and there is a pending interrupt:
10.18.1. Save intermediate state in the migration stream context.
10.18.2. Terminate TDH.EXPORT.MEM with a TDX_INTERRUPTED_RESUMABLE status.
10.19. Else, advance to the next entry in the GPA list, if applicable.
11. Once the GPA list has been fully processed, update the migration stream next MB counter field.

Completion Status Codes

Table 5.17: TDH.EXPORT.MEM Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	
TDX_INVALID_RESUMPTION	
TDX_INTERRUPTED_RESUMABLE	
TDX_MIGRATION_STREAM_STATE_INCORRECT	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	Operation is successful. **Note:** Processing of some GPA list entries may have encountered errors, but this did not cause an abort of the overall operation. The number such errors is reported in the lower 32 bits of the completion status.

Completion Status Code	Description
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.5. TDH.EXPORT.PAUSE Leaf

TDH.EXPORT.PAUSE starts the TDX-enforced blackout period on the source platform, where the source TD is paused.

Table 5.18: TDH.EXPORT.PAUSE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	HPA of Source TD TDR page (HKID bits must be 0)		

Table 5.19: TDH.EXPORT.PAUSE Output Operands Definitions

Operand	Description
RAX	SEAMCALL instruction return code, see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.EXPORT.PAUSE starts the Live Migration Blackout period on the source platform.

Enumeration: Availability of TDH.EXPORT.PAUSE is enumerated by either TDX_FEATURES0.TD_MIGRATION (bit 0) or TDX_FEATURES0.S4 (bit 13), readable by TDH.SYS.RD* (see 3.3.3.1), being set to 1. If not supported, calling TDH.EXPORT.PAUSE returns a TDX_OPERAND_INVALID(RAX) status.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.20: TDH.EXPORT.PAUSE Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	R	Opaque	4KB	Exclusive	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	TDCS Epoch Tracking Fields	N/A	RW	Opaque	N/A	Exclusive(i)	N/A	N/A

TDH.EXPORT.PAUSE checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. TDCS.OP_STATE is LIVE_EXPORT.

Note: All TD VCPUs have stopped executing and no other TD-specific SEAMCALL is running. This is implicit, since TDH.EXPORT.PAUSE has an exclusive access to TDR and TDCS.

If successful, the function does the following:

6. Increment the TD's epoch counter (TDCS.TD_EPOCH).

Note: This allows memory management operations to skip the need for blocking and TLB tracking while the TD is paused. If the export session is aborted, the first TDH.VP.ENTER on each VCPU will flush TLB.

7. Set the TDCS.OP_STATE to PAUSED_EXPORT.

Completion Status Codes

Table 5.21: TDH.EXPORT.PAUSE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.6. TDH.EXPORT.RESTORE Leaf

TDH.EXPORT.RESTORE restores a list of TD private 4KB pages' Secure EPT entry states after an export abort.

Table 5.22: TDH.EXPORT.RESTORE Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	SEAMCALL instruction leaf number and version, see 5.4.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the SEAMCALL interface function
		23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
		63:24	Reserved	Must be 0
RCX	GPA_LIST_INFO	GPA_LIST_INFO: HPA of a GPA list page in shared memory, and first and last entries to process, as defined in 3.12.2 FORMAT must be GPA_ONLY.		
RDX	TDR	HPA of the source TD's TDR page (HKID bits must be 0)		

Table 5.23: TDH.EXPORT.RESTORE Output Operands Definition

Operand	Name	Description
RAX	STATUS	SEAMCALL instruction return code, see 5.4.1
RCX	GPA_LIST_INFO	Same as the input value, except that FIRST_ENTRY is updated to the index of the next entry to be processed. If all entries have been processed, FIRST_ENTRY is updated to (LAST_ENTRY + 1) Modulo 512.
Other		Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.EXPORT.RESTORE restores a list of TD private 4KB pages' Secure EPT entry states after an aborted export session. It reverts each L1 Secure EPT entry and any applicable L2 Secure EPT entries to their original non-exported state.

Enumeration: Availability of TDH.EXPORT.RESTORE is enumerated by TDX_FEATURES0.TD_MIGRATION (bit 0), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.EXPORT.RESTORE returns a TDX_OPERAND_INVALID(RAX) status.

List Entry Error: If a page's Secure EPT entry can't be restored, TDH.EXPORT.RESTORE marks its GPA list entry as unsuccessful. List process is not aborted; it continues to the next entry, if applicable. The return status in RAX indicates the number of such cases encountered during operation.

Interruptibility: TDH.EXPORT.RESTORE is interruptible. If a pending interrupt is detected during operation, TDH.EXPORT.RESTORE returns with a TDX_INTERRUPTED_RSUMABLE status in RAX. RCX is updated with the next list entry index to process, so the host VMM may re-invoke TDH.EXPORT.RESTORE immediately after handling the interrupt.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.24: TDH.EXPORT.RESTORE Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	GPA List page	GPA_LIST	RW	Shared	4KB	None	None	None
Explicit	RDX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Explicit	N/A	GPA	TD private pages (via GPA list)	Block	None	Private	4KB	None	None	None
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	R	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	None	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive(h)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT trees	N/A	RW	Private	N/A	Shared(i)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A

TDH.EXPORT.RESTORE checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. TDCS.OP_STATE is RUNNABLE.

If passed, process the GPA list:

Note: Error conditions that impact a single GPA list entry do not cause an abort of TDH.EXPORT.RESTORE. Instead, the GPA list entry is updates with a proper status code, and the corresponding migration buffer list entry is marked as invalid.

6. For each entry in the GPA list, starting with RCX.FIRST_ENTRY and ending with RCX.LAST_ENTRY, if OPERATION indicates a RESTORE request:
   6.1. Check the GPA list entry fields value.

   If passed:

   6.2. Walk the L1 SEPT based on the GPA and level operands and find the leaf entry for the page.
   6.3. Check that the SEPT entry state is one of the EXPORTED_* or PENDING_EXPORTED_* states.
   6.4. If passed, update the SEPT entry:
      6.4.1. Atomically decrement TDCS.MIG_COUNT.
      6.4.2. If the SEPT state is one of the *_DIRTY* states, atomically decrement TDCS.DIRTY_COUNT.
      6.4.3. If the SEPT state is one of the PENDING_* states, update it to PENDING. Else, update it to MAPPED.
      6.4.4. If the page has any L2 mappings, and the SEPT state was one of the non-PENDING but BLOCKEDW states, then for each L2 SEPT:

6.6.1.1. Walk the L2 SEPT tree based on the GPA operand and find the Secure EPT entry to be blocked.
6.6.1.2. If found:
6.6.1.2.1. Save the original value of SEPT.W into SEPT.TDW.
6.6.1.2.2. Clear SEPT.W.

6.5. Else:
6.5.1. Set the GPA list entry's OPERATION field to NOP and STATUS field to the applicable status.

6.6. If this is not the last entry in the list, and there is a pending interrupt, terminate TDH.EXPORT.RESTORE with a TDX_INTERRUPTED_RESUMABLE status.

Completion Status Codes

Table 5.25: TDH.EXPORT.RESTORE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	Operation is successful. **Note:** Processing of some GPA list entries may have encountered errors, but this did not cause an abort of the overall operation. The number such errors is reported in the lower 32 bits of the completion status.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.7. TDH.EXPORT.STATE.IMMUTABLE Leaf

TDH.EXPORT.STATE.IMMUTABLE starts a new export session and exports the TD's immutable state as a multi-page migration bundle.

TDH.EXPORT.STATE.IMMUTABLE is also used for starting a new S4 hibernation session.

Table 5.26: TDH.EXPORT.STATE.IMMUTABLE Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	SEAMCALL instruction leaf number and version, see 5.4.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the SEAMCALL interface function
		23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
		63:24	Reserved	Must be 0
RCX	TDR	Source TD handle and flags		
		Bits	**Name**	**Description**
		0	EXPORT_TYPE	0: TD Export 1: S4 Hibernation
		11:1	Reserved	Must be 0
		51:12	TDR HPA	HPA[51:12] of the source TD's TDR page (HKID bits must be 0)
		63:52	Reserved	Must be 0
R8	MBMD	HPA and size of memory of a memory buffer to use for MBMD:		
		Bits	**Name**	**Description**
		51:0	HPA	Bits 51:0 of the host physical address (including HKID bits)
		63:52	Size	Size of the memory buffer containing MBMD, in bytes
R9	PAGE_LIST_INFO	Migration buffers list information – see 3.12.6.1		
R10	MIG_STREAM	Migration stream and resume flag:		
		Bits	**Name**	**Description**
		15:0	MIGS_INDEX	Migration stream index – must be 0
		62:16	RESERVED	Reserved: must be 0
		63	RESUME	0: This is a new invocation 1: This is resumption of a previously interrupted operation

Table 5.27: TDH.EXPORT.STATE.IMMUTABLE Output Operands Definition

Operand	Name	Description
RAX	STATUS	SEAMCALL instruction return code, see 5.4.1
RDX	NUM_EXPORTED	Number of exported 4KB migration buffers
AVX, AVX2 and AVX512 state		May be reset to the architectural RESET state
Other		Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.EXPORT.STATE.IMMUTABLE starts a new export session. It exports the TD's immutable state as a migration bundle, which includes an MBMD and a set of 4KB pages, encrypted with the migration session key. The migration bundle is protected by a MAC that is stored in the MBMD.

TDH.EXPORT.STATE.IMMUTABLE is also used for starting a new S4 hibernation session.

Enumeration: Availability of TDH.EXPORT.STATE.IMMUTABLE is enumerated by either TDX_FEATURES0.TD_MIGRATION (bit 0) or TDX_FEATURES0.S4 (bit 13), readable by TDH.SYS.RD* (see 3.3.3.1), being set to 1. If not supported, calling TDH.EXPORT.STATE.IMMUTABLE returns a TDX_OPERAND_INVALID(RAX) status.

Interruptibility: TDH.EXPORT.STATE.IMMUTABLE is interruptible. The host VMM is expected to invoke it in a loop until it returns with either a success indication or with a non-recoverable error indication.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.28: TDH.EXPORT.STATE.IMMUTABLE Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Explicit	R8	HPA	Memory to use for MBMD	MBMD	RW	Shared	128B	None	None	None
Explicit	R9	HPA	Page list	PAGE_LIST	RW	Shared	4KB	None	None	None
Explicit	R10	N/A	Mig. Stream context	Mig. Stream context	RW	Opaque	N/A	Exclusive	N/A	N/A
Explicit	N/A	HPA	Destination pages (via page list)	Blob	RW	Shared	4KB	None	None	None
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	Exclusive	N/A	N/A

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	N/A	PL.S4_STATE	N/A	RW	Hidden	N/A	Exclusive	N/A	N/A

TDH.EXPORT.STATE.IMMUTABLE checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. The TD build and measurement have been finalized, or the TD has been imported, and no export session is in progress (TDCS.OP_STATE is either RUNNABLE or LIVE_IMPORT).
6. The TD is migratable: TDCS.ATTRIBUTES.MIGRATABLE is set to 1.
   6.1. For S4:
      6.1.1. This check is only applicable if EXPORT_TYPE is 0 (TD export).
      6.1.2. If TD migration is not supported, then EXPORT_TYPE must not be 0 (TD export).
7. Any previous aborted export session has been cleaned up: TDCS.MIG_COUNT is 0.
8. MIGS_INDEX is 0.
9. The buffer provided for MBMD is large enough.
10. The number of pages in the page list is large enough to hold the exported state.

    Note: The required number of pages is enumerated by TDH.SYS.RD*.

If successful, the function does the following:

11. If the RESUME input flag is 0, indicating that this is a new invocation of TDH.EXPORT.STATE.IMMUTABLE (not a resumption of a previously interrupted one):
    11.1. If EXPORT_TYPE is 0 (TD export):
       11.1.1. Check that a valid migration decryption key has been set by the Migration TD. If this is not the first migration session, then the migration key must have been set after the previous migration session has started.

          Note: There is no explicit check that a migration TD is bound; this is implied by the above check.

       11.1.2. For S4, set TDCS.S4_MIGRATED to FALSE.

    11.2. Else (EXPORT_TYPE is 1 (S4 hibernation) – only if S4 is supported):
       11.2.1. Check that PL.S4_STATE is either S4_IDLE or S4_EXPORT.
       11.2.2. Create the S4 encryption key and set TDCS.MIG_ENC_WORKING_KEY to this value.
       11.2.3. Set TDCS.MIG_WORKING_VERSION to S4_MIG_VERSION (an TDX module constant).
       11.2.4. If the global S4 session has not yet started, start it (set PL.S4_STATE to S4_EXPORT).
       11.2.5. Set TDCS.S4_MIGRATED to TRUE.
       11.2.6. Atomically increment PL.S4_EXP_INDEX.

    If passed:

    11.3. Initialize the migration context in TDCS:
       11.3.1. Copy the migration keys to working migration keys that will be used throughout the export session.

       If passed:

       11.3.2. Set all migration streams' INITIALIZED flags to 0 and ENABLED flags to 1.
    11.4. Initialize the current migration stream.
    11.5. Increment the migration stream context's IV_COUNTER.
    11.6. Build the 96b IV for this migration bundle by concatenating 0 as the direction bit, the stream index and the stream context's IV_COUNTER.

11.7. Build the MBMD in the migration stream context.
11.8. Accumulate MAC in the stream context based on the MAC'ed fields of MBMD.
12. Else (this is a resumption of a previously interrupted TDH.EXPORT.STATE.IMMUTABLE):
12.1. Check that the resumption is valid:
12.1.1. The stream context indicates there's a valid interruption state.
12.1.2. The current SEAMCALL leaf number and the PAGE_OR_LIST operand have the same value as in the interruption state.
12.2. Check that the migration stream is enabled.
12.3. Restore the previously saved page list index from the migration context.
12.4. Build the 96b IV for this migration bundle by concatenating the stream index and the stream context's IV_COUNTER.

If passed:

13. Repeat exporting 4KB pages until all immutable state is exported or until a pending interrupt is detected:
13.1. Get the 4KB next page HPA from it from the page list.
13.2. Dump the next set of metadata fields as a metadata list of field sequences, into an internal temporary 4KB buffer.
13.3. Use the migration key and the migration stream context to encrypt the 4KB internal buffer into the destination data page and update the MAC calculation.
13.4. If all immutable state has been exported:
13.4.1. Write the accumulated MAC to the MBMD in the stream context.
13.4.2. Write the MBMD to the memory buffer provided by the host VMM.
13.4.3. Mark the migration stream context's interrupted state as invalid.
13.4.4. Increment the migration stream context's NEXT_MB_COUNTER.
13.4.5. Set TDCS.TOTAL_MB to 1.
13.4.6. Set TDCS.OP_STATE to LIVE_EXPORT.
13.4.7. Clear TDCS.DIRTY_COUNT to 0.
13.4.8. Terminate TDH.EXPORT.STATE.IMMUTABLE with a TDX_SUCCESS status.
13.5. Else, if there is a pending interrupt:
13.5.1. Save the interruption state to the stream context
13.5.2. Terminate TDH.EXPORT.STATE.IMMUTABLE with a TDX_INTERRUPTED_RESUMABLE status.

Completion Status Codes

Table 5.29: TDH.EXPORT.STATE.IMMUTABLE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	
TDX_INVALID_RESUMPTION	
TDX_IOMMU_IOTLB_TRACKING_NOT_DONE	Applicable only if TDX Connect is supported
TDX_MAX_EXPORTS_EXCEEDED	
TDX_METADATA_LIST_OVERFLOW	
TDX_MIGRATION_SESSION_KEY_NOT_SET	
TDX_MIGRATION_STREAM_STATE_INCORRECT	
TDX_MIN_MIGS_NOT_CREATED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	

Completion Status Code	Description
TDX_PREVIOUS_EXPORT_CLEANUP_INCOMPLETE	
TDX_RND_NO_ENTROPY	Failed to generate a random migration encryption key. This is typically caused by an entropy error of the CPU's random number generator, and may be impacted by RDSEED, RDRAND or PCONFIG executing on other LPs. The operation should be retried.
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_HAS_ATTACHED_DEVICES	Applicable only if TDX Connect is supported
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_MIGRATABLE	
TDX_TDCS_NOT_ALLOCATED	

5.4.8. TDH.EXPORT.STATE.TD Leaf

TDH.EXPORT.STATE.TD exports a paused TD's mutable state as a multi-page migration bundle.

Table 5.30: TDH.EXPORT.STATE.TD Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	SEAMCALL instruction leaf number and version, see 5.4.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the SEAMCALL interface function
		23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
		63:24	Reserved	Must be 0
RCX	TDR	HPA of the source TD's TDR page (HKID bits must be 0)		
R8	MBMD	HPA and size of memory of a memory buffer to use for MBMD:		
		Bits	**Name**	**Description**
		51:0	HPA	Bits 51:0 of the host physical address (including HKID bits)
		63:52	Size	Size of the memory buffer containing MBMD, in bytes
R9	PAGE_LIST_INFO	Migration buffers list information – see 3.12.6.1		
R10	MIG_STREAM	Migration stream and resume flag:		
		Bits	**Name**	**Description**
		15:0	MIGS_INDEX	Migration stream index – must be 0
		62:16	RESERVED	Reserved: must be 0
		63	RESUME	0: This is a new invocation 1: This is resumption of a previously interrupted operation

Table 5.31: TDH.EXPORT.STATE.TD Output Operands Definition

Operand	Name	Description
RAX	STATUS	SEAMCALL instruction return code, see 5.4.1
RDX	NUM_EXPORTED	Number of exported 4KB migration buffers
AVX, AVX2 and AVX512 state		May be reset to the architectural RESET state
Other		Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.EXPORT.STATE.TD exports the TD's mutable state as a migration bundle, which includes an MBMD and a set of 4KB pages, encrypted with the migration session key. The migration bundle is protected by a MAC that is stored in the MBMD. The TD must have been paused by a TDH.EXPORT.PAUSE.

Enumeration: Availability of TDH.EXPORT.STATE.TD is enumerated by either TDX_FEATURES0.TD_MIGRATION (bit 0) or TDX_FEATURES0.S4 (bit 13), readable by TDH.SYS.RD* (see 3.3.3.1), being set to 1. If not supported, calling TDH.EXPORT.STATE.TD returns a TDX_OPERAND_INVALID(RAX) status.

Interruptibility: TDH.EXPORT.STATE.TD is interruptible. The host VMM is expected to invoke it in a loop until it returns with either a success indication or with a non-recoverable error indication.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.32: TDH.EXPORT.STATE.TD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Explicit	R8	HPA	Memory to use for MBMD	MBMD	RW	Shared	128B	None	None	None
Explicit	R9	HPA	Page list	PAGE_LIST	R	Shared	4KB	None	None	None
Explicit	R10	Index	Mig. Stream context	Mig. Stream context	RW	Opaque	N/A	Exclusive	N/A	N/A
Explicit	N/A	HPA	Destination pages (via page list)	Blob	RW	Shared	4KB	None	None	None
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	R	Opaque	N/A	Exclusive(h)	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	None	N/A	N/A

TDH.EXPORT.STATE.TD checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. An export session is in progress, and the TD has been paused: TDCS.OP_STATE is PAUSED_EXPORT.
6. Migration stream index is 0.
7. The migration stream is enabled and initialized.
8. The buffer provided for MBMD is large enough.
9. The number of pages in the page list is large enough to hold the exported state.

Note: The required number of pages is enumerated by TDH.SYS.RD*.

If successful, the function does the following:

10. If the RESUME input flag is 0, indicating that this is a new invocation of TDH.EXPORT.STATE.TD (not a resumption of a previously interrupted one):
    10.1. Increment the migration stream context's IV_COUNTER.

10.2. Build the 96b IV for this migration bundle by concatenating 0 as the direction bit, the stream index (0) and the stream context's IV_COUNTER.
10.3. Build the MBMD in the migration stream context.
10.4. Accumulate MAC in the stream context based on the MAC'ed fields of MBMD.
11. Else (this is a resumption of a previously interrupted TDH.EXPORT.STATE.TD):
11.1. Check that the resumption is valid:
11.1.1. The stream context indicates there's a valid interruption state.
11.1.2. The current SEAMCALL leaf number and the PAGE_OR_LIST operand have the same value as in the interruption state.
11.2. Build the 96b IV for this migration bundle by concatenating the stream index and the stream context's IV_COUNTER.
11.3. Restore the previously saved page list index from the migration context.

If passed:

12. Repeat exporting 4KB pages until all mutable TD state is exported or until a pending interrupt is detected:
12.1. Get the 4KB next page HPA from it from the page list.
12.2. Dump the next set of metadata fields as a metadata list of field sequences, into an internal temporary 4KB buffer.
12.3. Use the migration key and the migration stream context to encrypt the 4KB internal buffer into the destination data page and update the MAC calculation.
12.4. If all TD state has been exported:
12.4.1. Write the accumulated MAC to the MBMD in the stream context.
12.4.2. Write the MBMD to the memory buffer provided by the host VMM.
12.4.3. Mark the migration stream context's interrupted state as invalid.
12.4.4. Increment the migration stream context's NEXT_MB_COUNTER.
12.4.5. Increment TDCS.TOTAL_MB.
12.4.6. Terminate TDH.EXPORT.STATE.TD with a TDX_SUCCESS status.
12.5. Else, if there is a pending interrupt:
12.5.1. Save the interruption state to the stream context
12.5.2. Terminate TDH.EXPORT.STATE.TD with a TDX_INTERRUPTED_RESUMABLE status.

Completion Status Codes

Table 5.33: TDH.EXPORT.STATE.TD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	
TDX_INVALID_RESUMPTION	
TDX_METADATA_LIST_OVERFLOW	
TDX_MIGRATION_STREAM_STATE_INCORRECT	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	

Completion Status Code	Description
TDX_TDCS_NOT_ALLOCATED	

5.4.9. TDH.EXPORT.STATE.VP Leaf

TDH.EXPORT.STATE.VP exports a paused TD's VCPU mutable state as a multi-page migration bundle.

Table 5.34: TDH.EXPORT.STATE.VP Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	SEAMCALL instruction leaf number and version, see 5.4.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the SEAMCALL interface function
		23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
		63:24	Reserved	Must be 0
RCX	TDVPR	HPA of the source TD VCPU's TDVPR page (HKID bits must be 0)		
R8	MBMD	HPA and size of memory of a memory buffer to use for MBMD:		
		Bits	**Name**	**Description**
		51:0	HPA	Bits 51:0 of the host physical address (including HKID bits)
		63:52	Size	Size of the memory buffer containing MBMD, in bytes
R9	PAGE_LIST_INFO	Migration buffers list information – see 3.12.6.1		
R10	MIG_STREAM	Migration stream and resume flag:		
		Bits	**Name**	**Description**
		15:0	MIGS_INDEX	Migration stream index If N migration streams have been created by TDH.MIG.STREAM.CREATE, then MIGS_INDEX must be lower than N-1.
		62:16	RESERVED	Reserved: must be 0
		63	RESUME	0: This is a new invocation 1: This is resumption of a previously interrupted operation

Table 5.35: TDH.EXPORT.STATE.VP Output Operands Definition

Operand	Name	Description
RAX	STATUS	SEAMCALL instruction return code, see 5.4.1
RDX	NUM_EXPORTED	Number of exported 4KB migration buffers
AVX, AVX2 and AVX512 state		May be reset to the architectural RESET state
Other		Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.EXPORT.STATE.VP exports a TD's VCPU mutable state as a migration bundle, which includes an MBMD and a set of 4KB pages, encrypted with the migration session key. The migration bundle is protected by a MAC that is stored in the MBMD. The TD must have been paused by a TDH.EXPORT.PAUSE.

Enumeration: Availability of TDH.EXPORT.STATE.VP is enumerated by either TDX_FEATURES0.TD_MIGRATION (bit 0) or TDX_FEATURES0.S4 (bit 13), readable by TDH.SYS.RD* (see 3.3.3.1), being set to 1. If not supported, calling TDH.EXPORT.STATE.VP returns a TDX_OPERAND_INVALID(RAX) status.

Interruptibility: TDH.EXPORT.STATE.VP is interruptible. The host VMM is expected to invoke it in a loop until it returns with either a success indication or with a non-recoverable error indication.

VCPU Association: TDH.EXPORT.VP associates the TD VCPU with the current LP. This requires that the VCPU will not be associated with another LP – for details, see the [TDX Module Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.36: TDH.EXPORT.STATE.VP Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDVPR page	TDVPS	RW	Opaque	4KB	Exclusive	Shared	Shared
Explicit	R8	HPA	Memory to use for MBMD	MBMD	RW	Shared	128B	None	None	None
Explicit	R9	HPA	Page list	PAGE_LIST	R	Shared	4KB	None	None	None
Explicit	R10	Index	Mig. Stream context	Mig. Stream context	RW	Opaque	N/A	Exclusive	N/A	N/A
Explicit	N/A	HPA	Destination pages (via page list)	Blob	RW	Shared	4KB	None	None	None
Implicit	N/A	HPA	TDR page	TDR	R	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	R	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	None	N/A	N/A

TDH.EXPORT.STATE.VP checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDVPR page metadata in PAMT must be correct (PT must be PT_TDVPR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. An export session is in progress, and the TD has been paused: TDCS.OP_STATE is PAUSED_EXPORT.
6. Migration stream index is lower than TDCS.NUM_MIGS.
7. The migration stream is enabled.
8. The buffer provided for MBMD is large enough.

9. The number of pages in the page list is large enough to hold the exported state.

   Note: The required number of pages is enumerated by TDH.SYS.RD*.

If successful, the function does the following:

10. Associate the VCPU with the current LP, and update TD VMCS using the algorithm described in 5.3.1.

If passed:

11. If the RESUME input flag is 0, indicating that this is a new invocation of TDH.EXPORT.STATE.VP (not a resumption of a previously interrupted one):
    11.1. If the migration stream has not been initialized, initialize it.
    11.2. Increment the migration stream context's IV_COUNTER.
    11.3. Build the MBMD in the migration stream context.
    11.4. Build the 96b IV for this migration bundle by concatenating the stream index and the stream context's IV_COUNTER.
    11.5. Accumulate MAC in the stream context based on the MAC'ed fields of MBMD.
12. Else (this is a resumption of a previously interrupted TDH.EXPORT.STATE.VP):
    12.1. Check that the resumption is valid:
        12.1.1. The stream context indicates there's a valid interruption state.
        12.1.2. The current SEAMCALL leaf number, and the TDVPR HPA and PAGE_OR_LIST operands are the same as in the interruption state.
    12.2. Increment the migration stream context's IV_COUNTER.
    12.3. Restore the previously saved page list index from the migration context.
13. Repeat exporting 4KB pages until all immutable state is exported or until a pending interrupt is detected:
    13.1. Get the 4KB next page HPA from it from the page list.
    13.2. Dump the next set of metadata fields as a metadata list of field sequences, into an internal temporary 4KB buffer.
    13.3. Use the migration key and the migration stream context to encrypt the 4KB internal buffer into the destination data page and update the MAC calculation.
    13.4. If all VCPU state has been exported:
        13.4.1. Write the accumulated MAC to the MBMD in the stream context.
        13.4.2. Write the MBMD to the memory buffer provided by the host VMM.
        13.4.3. Mark the migration stream context's interrupted state as invalid.
        13.4.4. Increment the migration stream context's NEXT_MB_COUNTER.
        13.4.5. Increment TDCS.TOTAL_MB.
        13.4.6. Terminate TDH.EXPORT.STATE.VP with a TDX_SUCCESS status.
    13.5. Else, if there is a pending interrupt:
        13.5.1. Save the interruption state to the stream context
        13.5.2. Terminate TDH.EXPORT.STATE.VP with a TDX_INTERRUPTED_RESUMABLE status.

Completion Status Codes

Table 5.37: TDH.EXPORT.STATE.VP Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	
TDX_INVALID_RESUMPTION	
TDX_METADATA_LIST_OVERFLOW	
TDX_MIGRATION_STREAM_STATE_INCORRECT	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	

Completion Status Code	Description
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	
TDX_VCPU_ALREADY_EXPORTED	

5.4.10. TDH.EXPORT.TRACK Leaf

TDH.EXPORT.TARCK ends the current in-order export phase epoch and either starts a new epoch or starts the out-of-order export phase. Generate an epoch token to be exported to the destination platform.

Table 5.38: TDH.EXPORT.TRACK Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	HPA of the source TD's TDR page (HKID bits must be 0)		
R8	HPA and size of memory of a memory buffer to use for MBMD:		
	Bits	**Name**	**Description**
	51:0	HPA	Bits 51:0 of the host physical address (including HKID bits)
	63:52	Size	Size of the memory buffer containing MBMD, in bytes
R10	Migration stream and flags:		
	Bits	**Name**	**Description**
	15:0	MIGS_INDEX	Migration stream index – must be 0
	62:16	RESERVED	Reserved: must be 0
	63	IN_ORDER_DONE	Indicates that the in-order export phase is done

Table 5.39: TDH.EXPORT.TRACK Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 5.4.1
AVX, AVX2 and AVX512 state	May be reset to the architectural RESET state
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

Enumeration: Availability of TDH.EXPORT.TRACK is enumerated by either TDX_FEATURES0.TD_MIGRATION (bit 0) or TDX_FEATURES0.S4 (bit 13), readable by TDH.SYS.RD* (see 3.3.3.1), being set to 1. If not supported, calling TDH.EXPORT.TRACK returns a TDX_OPERAND_INVALID(RAX) status.

IN_ORDER_DONE: If R10.IN_ORDER_DONE is 0, TDH.EXPORT.TRACK starts a new export epoch.

Else (R10.IN_ORDER_DONE is 1), TDH.EXPORT.TRACK checks that no memory exported so far needs to be re-exported. If so, it ends the in-order export phase and starts the out-of-order phase.

In both cases, TDH.EXPORT.TRACK generates an epoch token, to be exported on the specified migration stream.

When called as part of S4 hibernation, R10.IN_ORDER_DONE must be 1.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.40: TDH.EXPORT.TRACK Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Explicit	R8	HPA	Memory to use for MBMD	MBMD	RW	Shared	128B	None	None	None
Explicit	R10	Index	Mig. Stream context	Mig. Stream context	RW	Opaque	N/A	Exclusive	N/A	N/A
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	Exclusive	N/A	N/A

TDH.EXPORT.TRACK checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. An export session is in the in-order phase: TDCS.OP_STATE is either LIVE_EXPORT or PAUSED_EXPORT.
6. The migration stream index is 0.
7. The migration stream is initialized.
8. The buffer provided for MBMD is large enough.

If successful, the function does the following:

9. If called as part of S4 hibernation, check that R10.IN_ORDER_DONE is 1.

If passed:

10. If R10.IN_ORDER_DONE is 0:
    10.1. Increment TDCS.MIG_EPOCH
11. Else (R10.IN_ORDER_DONE is 1):
    11.1. Check that an export session is in the in-order phase and the TD has been paused: TDCS.OP_STATE is PAUSED_EXPORT.
    11.2. Check that TDCS.DIRTY_COUNT is 0, indicating that no unexported newer versions of any memory page exported so far remain. Memory pages that have not yet been exported may remain and may later be exported (out-of-order).

If passed:

11.3. Start the out-of-order phase:
    11.3.1. Set TDCS.OP_STATE to POST_EXPORT.
    11.3.2. Set TDCS.MIG_EPOCH to 0xFFFFFFFF.
12. Increment the migration stream context's IV_COUNTER.
13. Build the 96b IV for this migration bundle by concatenating the stream index and the stream context's IV_COUNTER.
14. Create an epoch token MBMD with the following fields:
    14.1. The number of the new epoch that have just begun. Bit 63 indicates the beginning of the out-of-order phase.
    14.2. The total number of migration bundles (including the current one) that have been exported in the current migration session.
15. Accumulate MAC based on the MAC'ed fields of MBMD and write to the MBMD's MAC field's value.
16. Write the MBMD to the provided memory buffer.

Completion Status Codes

Table 5.41: TDH.EXPORT.TRACK Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EXPORTED_DIRTY_PAGES_REMAIN	
TDX_MIGRATION_EPOCH_OVERFLOW	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.11. TDH.EXPORT.UNBLOCKW Leaf

Remove the write-blocking of a 4KB TD private page previously blocked by TDH.EXPORT.BLOCKW.

Table 5.42: TDH.EXPORT.UNBLOCKW Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the Secure EPT entry that maps the page to be blocked for writing – see 3.6.1: must be 0 (4KB)
	11:3	Reserved	Reserved: must be 0
	51:12	GPA	Bits 51:12 of the GPA to be unblocked for writing
	63:52	Reserved	Reserved: must be 0
RDX	Host physical address of the parent TDR page (HKID bits must be 0)		

Table 5.43: TDH.EXPORT.UNBLOCKW Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 5.4.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry where the error was detected In other cases, RCX returns 0
RDX	Extended error information part 2 In case of EPT walk error, EPT level where the error was detected In other cases, RDX returns 0
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.EXPORT.UNBLOCKW finds the write blocked Secure EPT entry for the given GPA and level. It verifies that the entry has been blocked for writing and TLB tracking has been done, then marks the entry as non-blocked for writing (MAPPED, PENDING, EXPORTED_DIRTY or PENDING_EXPORTED_DIRTY as appropriate). If the page has any L2 mappings, TDH.EXPORT.UNBLOCKW unblocks them.

Enumeration: Availability of TDH.EXPORT.UNBLOCKW is enumerated by TDX_FEATURES0.TD_MIGRATION (bit 0), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.EXPORT.UNBLOCKW returns a TDX_OPERAND_INVALID(RAX) status.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.44: TDH.EXPORT.UNBLOCKW Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA and Level	Secure EPT page or TD private page	Blob	None	Private	$2^{12+9*Level}$ Bytes	None	None	None
Explicit	RDX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	R	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	None	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive(h)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT trees	N/A	RW	Private	N/A	Shared(i)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A

TDH.EXPORT.UNBLOCKW checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. Either of the following is true:
   5.1. An export session is in progress.
   5.2. The TD is allowed to run (TDCS.OP_STATE is either RUNNABLE, LIVE EXPORT, PAUSED_EXPORT or POST_EXPORT). In these states, TDH.EXPORT.UNBLOCKW is used to clean up after an aborted export session.
6. The specified level is 0 (4KB).

If successful, the function does the following:

7. Walk the Secure EPT based on the GPA operand and find the Secure EPT page or TD private page to be unblocked for writing.
8. Check the Secure EPT entry state is blocked for writing: BLOCKEDW, PENDING_BLOCKEDW, EXPORTED_DIRTY_BLOCKEDW or PENDING_EXPORTED_DIRTY_BLOCKEDW.
9. If the TD is allowed to run, check that TLB tracking was done.

If passed:

10. If the page state is not one of the PENDING* states:
    10.1. Restore the original value of SEPT.W from SEPT.TDW.
11. If the page has not been exported (Secure EPT entry state is BLOCKEDW or PENDING_BLOCKEDW), unblock the Secure EPT entry for writing by atomically setting its state to MAPPED or PENDING, respectively.
12. Else (Secure EPT entry state is EXPORTED_DIRTY_BLOCKEDW or PENDING_EXPORTED_DIRTY_BLOCKEDW):
    12.1. Unblock the Secure EPT entry for writing by atomically setting its state to EXPORTED_DIRTY or PENDING_EXPORTED_DIRTY, respectively.
    12.2. Atomically increment TDCS.DIRTY_COUNT.
13. If the updated page state is MAPPED or EXPORTED_DIRTY, then for each L2 mapping of the page:
    13.1. Walk the L2 SEPT tree based on the GPA operand and find the L2 Secure EPT entry to be unblocked for writing.
    13.2. Restore the original value of SEPT.W from SEPT.TDW.
    13.3. Set the L2 SEPT entry state to L2_MAPPED

Note: If the updated page state is one of the PENDING* states, then the L2 SEPT entry state is already L2_BLOCKED, no change is required.

Completion Status Codes

Table 5.45: TDH.EXPORT.UNBLOCKW Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_STATE_INCORRECT	
TDX_EPT_WALK_FAILED	
TDX_NOT_WRITE_BLOCKED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.EXPORT.UNBLOCKW is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	
TDX_TLB_TRACKING_NOT_DONE	

5.4.12. TDH.IMPORT.ABORT Leaf

Abort an import session; after this the target TD can only be destroyed. Generate an abort token that is to be consumed by the source platform.

Table 5.46: TDH.IMPORT.ABORT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	HPA of the source TD's TDR page (HKID bits must be 0)		
R8	HPA and size of memory of a memory buffer to use for MBMD:		
	Bits	**Name**	**Description**
	51:0	HPA	Bits 51:0 of the host physical address (including HKID bits)
	63:52	Size	Size of the memory buffer containing MBMD, in bytes
R10	Migration stream index – must be 0		

Table 5.47: TDH.IMPORT.ABORT Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 5.4.1
AVX, AVX2 and AVX512 state	May be reset to the architectural RESET state
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.IMPORT.ABORT generates an abort token MBMD and sets the destination TD's OP_STATE to IMPORT_FAILED. In this state, the destination TD will not run; it can only be destroyed. This is indicated by the FATAL bit (61) of the completion status returned in RAX.

Enumeration: Availability of TDH.IMPORT.ABORT is enumerated by TDX_FEATURES0.TD_MIGRATION (bit 0), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.IMPORT.ABORT returns a TDX_OPERAND_INVALID(RAX) status.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.48: TDH.IMPORT.ABORT Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	R	Opaque	4KB	Exclusive	Shared	Shared
Explicit	R8	HPA	Memory to use for MBMD	MBMD	RW	Shared	128B	None	None	None
Explicit	R10	Index	Mig. Stream context	Mig. Stream context	RW	Opaque	N/A	Exclusive	N/A	N/A
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	Exclusive(i)	N/A	N/A

TDH.IMPORT.ABORT checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. An import session is in progress but has not been committed yet (TDCS.OP_STATE is one of MEMORY_IMPORT, STATE_IMPORT, POST_IMPORT or FAILED_IMPORT).
6. The migration stream index is 0.
7. The buffer provided for MBMD is large enough.

If successful, the function does the following:

8. Set TDCS.OP_STATE to FAILED_IMPORT.
9. If the migration stream has not been initialized, initialize it.
10. Increment the stream context's IV_COUNTER.
11. Build the 96b IV for this migration bundle by concatenating the stream index and the stream context's IV_COUNTER.
12. Create an abort token MBMD.
13. Accumulate MAC based on the MAC'ed fields of MBMD and write to the MBMD's MAC field's value.
14. Write the MBMD to the provided memory buffer.
15. Increment the stream context's NEXT_MB_COUNTER.

Completion Status Codes

Table 5.49: TDH.IMPORT.ABORT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	

Completion Status Code	Description
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

Intel TDX Application Binary Interface (ABI) Reference

5.4.13. TDH.IMPORT.COMMIT Leaf

Commit an import session and allow the imported TD to run.

Table 5.50: TDH.IMPORT.COMMIT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	HPA of the source TD's TDR page (HKID bits must be 0)		

Table 5.51: TDH.IMPORT.COMMIT Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.IMPORT.COMMIT commits an import session and allows the important TD to run. Post-copy memory import may continue.

Enumeration: Availability of TDH.IMPORT.COMMIT is enumerated by TDX_FEATURES0.TD_MIGRATION (bit 0), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.IMPORT.COMMIT returns a TDX_OPERAND_INVALID(RAX) status.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.52: TDH.IMPORT.COMMIT Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	R	Opaque	4KB	Exclusive	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	Exclusive(i)	N/A	N/A

TDH.IMPORT.COMMIT checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. An import session is in the out-of-order phase: TDCS.OP_STATE is POST_IMPORT.

If successful, the function does the following:

6. Set TDCS.OP_STATE to LIVE_IMPORT.

Completion Status Codes

Table 5.53: TDH.IMPORT.COMMIT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.14. TDH.IMPORT.END Leaf

End an import session.

Table 5.54: TDH.IMPORT.END Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	HPA of the source TD's TDR page (HKID bits must be 0)		

Table 5.55: TDH.IMPORT.END Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.IMPORT.END ends an import session and allows the important TD to run (if not already allowed by TDH.IMPORT.COMMIT).

When called as part of an S4 resumption session, TDH.IMPORT.END must be called after no future replay is prevented by calling TDH.SYS.S4_END.

Enumeration: Availability of TDH.IMPORT.END is enumerated by either TDX_FEATURES0.TD_MIGRATION (bit 0) or TDX_FEATURES0.S4 (bit 13), readable by TDH.SYS.RD* (see 3.3.3.1), being set to 1. If not supported, calling TDH.IMPORT.END returns a TDX_OPERAND_INVALID(RAX) status.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.56: TDH.IMPORT.END Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	R	Opaque	4KB	Exclusive	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	Exclusive(i)	N/A	N/A

TDH.IMPORT.END checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. An import session is in the out-of-order phase.
6. For S4, if TDCS.S4_MIGRATED is TRUE then a global S4 session in not in progress (PL.S4_STATE is S4_IDLE).

If successful, the function does the following:

7. Set TDCS.OP_STATE to RUNNABLE.

Completion Status Codes

Table 5.57: TDH.IMPORT.END Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.15. TDH.IMPORT.MEM Leaf

TDH.IMPORT.MEM imports a list of TD private pages contents and/or cancellation requests based on a migration bundle in shared memory.

Table 5.58: TDH.IMPORT.MEM Input Operands Definition

<table>
<tr><th>Operand</th><th>Name</th><th colspan="3">Description</th></tr>
<tr><td rowspan="5">RAX</td><td rowspan="5">Leaf and Version</td><td colspan="3">SEAMCALL instruction leaf number and version, see 5.4.1</td></tr>
<tr><td>Bits</td><td>Field</td><td>Description</td></tr>
<tr><td>15:0</td><td>Leaf Number</td><td>Selects the SEAMCALL interface function</td></tr>
<tr><td>23:16</td><td>Version Number</td><td>Selects the SEAMCALL interface function version
Must be 0</td></tr>
<tr><td>63:24</td><td>Reserved</td><td>Must be 0</td></tr>
<tr><td>RCX</td><td>GPA_LIST_INFO</td><td colspan="3">HPA of a GPA list page in shared memory, and first and last entries to process, as defined in 3.12.2
On a new invocation, FIRST_ENTRY must be 0. On a resumed invocation, FIRST_ENTRY must be the index of the next GPA list entry to export.</td></tr>
<tr><td>RDX</td><td>TDR</td><td colspan="3">HPA of the destination TD’s TDR page (HKID bits must be 0)</td></tr>
<tr><td rowspan="4">R8</td><td rowspan="4">MBMD</td><td colspan="3">HPA and size of memory of an MBMD structure in memory:</td></tr>
<tr><td>Bits</td><td>Name</td><td>Description</td></tr>
<tr><td>51:0</td><td>HPA</td><td>Bits 51:0 of the host physical address (including HKID bits)</td></tr>
<tr><td>63:52</td><td>Size</td><td>Size of the memory buffer containing MBMD, in bytes</td></tr>
<tr><td>R9</td><td>MIG_BUFF_LIST</td><td colspan="3">HPA (including HKID bits) of a migration buffer list in shared memory, corresponding to the GPA list pointed by RCX – see 3.12.3.
No migration buffers are required for PENDING pages and for migration cancellation requests. The list entries for such pages are skipped.</td></tr>
<tr><td rowspan="5">R10</td><td rowspan="5">MIG_STREAM</td><td colspan="3">Migration stream and resume flag:</td></tr>
<tr><td>Bits</td><td>Name</td><td>Description</td></tr>
<tr><td>15:0</td><td>MIGS_INDEX</td><td>Migration stream index
If N migration streams have been created by TDH.MIG.STREAM.CREATE, then MIGS_INDEX must be lower than N-1.</td></tr>
<tr><td>62:16</td><td>RESERVED</td><td>Reserved: must be 0</td></tr>
<tr><td>63</td><td>RESUME</td><td>0: This is a new invocation
1: This is resumption of a previously interrupted operation</td></tr>
<tr><td>R11</td><td>MAC_LIST_0</td><td colspan="3">HPA (including HKID bits) of a MAC list in shared memory, corresponding to the first 256 entries of the GPA list pointed by RCX – see 3.12.3.
If GPA_LIST_INFO.FIRST_ENTRY >= 256, then MAC_LIST_0 is ignored.</td></tr>
</table>

Operand	Name	Description
R12	MAC_LIST_1	HPA (including HKID bits) of a MAC list in shared memory, corresponding to the last 256 entries of the GPA list pointed by RCX – see 3.12.3. If GPA_LIST_INFO.LAST_ENTRY < 256, then MAC_LIST_1 is ignored.
R13	PAGE_LIST	If in-place import is requested for all pages imported for the first-time in the current import session, or for the first-time after a previous import cancellation, R13 should be set to NULL_PA (all 1's). Otherwise, if some pages are to be imported in a non-in-place mode, R13 should be set to the HPA (including HKID bits) of a destination page list in shared memory, corresponding to the GPA list pointed by RCX – see 3.10.6. The page list allows selecting in-place or non-in-place import for each page imported for the first-time in the current import session, or for the first-time after a previous import cancellation: • To select in-place import, the page list entry's INVALID bit should be set to 1 (it is possible to set the whole entry to NULL_PA). • To select non-in-place import, the page list entry should be set to the HPA (including HKID) of the page to become a new TD private page.
R14	ATTRIB_LIST	If GPA_LIST_INFO.FORMAT is GPA_AND_L2_ATTR, then R14 contains the HPA (including HKID bits) of a page L2 attributes list in shared memory – see 3.12.4. Else, R14 is ignored. An ATTRIB_LIST is mandatory for exporting partitioned TDs (which contain one or more L2 VMs). **Enumeration:** Support of this feature is enumerated by TDX_FEATURES0.PARTITIONED_TD_MIGRATION, readable by TDH.SYS.RD* (see 3.3.3.1).

Table 5.59: TDH.IMPORT.MEM Output Operands Definition

Operand	Name	Description
RAX	STATUS	SEAMCALL instruction return code, see 5.4.1
RCX	GPA_LIST_INFO	Same as the input value, except that FIRST_ENTRY is updated to the index of the next entry to be processed. If all entries have been processed, FIRST_ENTRY is updated to (LAST_ENTRY + 1) Modulo 512.
AVX, AVX2 and AVX512 state		May be reset to the architectural RESET state
Other		Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.IMPORT.MEM imports a list of up to 512 TD private 4KB pages based on a migration bundle, which includes an MBMD, set of 4KB pages encrypted with the migration session key, a 4KB page containing the GPA and attributes list, an optional 4KB containing page attributes list, and two 4KB pages containing page MACs.

For each page in the migration bundle's GPA list, the requested operation may either be to import the page, to re-import a newer version of the page (after a previous import) or to cancel a previous page import. It is also possible to skip entries in the list by requesting no operation for specific entries. The GPA list format is described in 3.12.2.

Enumeration: Availability of TDH.IMPORT.MEM is enumerated by either TDX_FEATURES0.TD_MIGRATION (bit 0) or TDX_FEATURES0.S4 (bit 13), readable by TDH.SYS.RD* (see 3.3.3.1), being set to 1. If not supported, calling TDH.IMPORT.MEM returns a TDX_OPERAND_INVALID(RAX) status.

TDX_FEATURES0.PARTITIONED_TD_MIGRATION (bit 21) enumerates TDX module support of migrating partitioned TDs (which contain one or more L2 VMs).

Re-Import: Re-import is only allowed during the in-order import phase. The imported pages replace an older version of the same pages, as long as the SEPT entry state is compatible:

- If the old SEPT state is PENDING, it may be overwritten by a new version that is either PENDING or MAPPED.
- If the old SEPT state is MAPPED, it may be overwritten by a newer version that is MAPPED.

Page attributes (e.g., RWX etc.) of a new page version may be different than those of a previously imported version.

If the out-of-order import phase, the imported pages may not overwrite an older version of the same pages.

In-Place Import: First-time import of a page during the current import session, or following a previous import cancellation, may be done in-place; the same physical pages that are provided as input are converted to TD private pages. Alternatively, a list of 4KB pages to be used as the destination TD new private pages may be provided. In any case, either a migration buffer or a new page must be provided, even if the imported page is PENDING and no content is imported.

Re-import of a page is always done over the TD private page that holds the previously imported version.

Import Abort: In many cases, an error during import aborts the import session because the memory state of the imported TD can't be guaranteed to be correct.

If the import session has not been committed yet (by THD.IMPORT.COMMIT) and not yet entered the LIVE_IMPORT state where the TD is allowed to run, a failed TDH.IMPORT.MEM is considered fatal to the import session (except in cases where the imported TD state has not been modified). The target TD is marked as IMPORT_FAILED and, by design, will not run. This is indicated by the FATAL bit (61) of the completion status returned in RAX.

If the import session has been committed and the entered the LIVE_IMPORT state where the TD is allowed to run, then a failed TDH.IMPORT.MEM terminates the import session (except in cases where the imported TD state has not been modified) but does not impact the TD's ability to run. This is indicated by the FATAL bit (61) of the completion status returned in RAX.

S4 Resumption: If TDH.IMPORT.MEM is called as part of an S4 hibernation, it only supports the out-of-order import phase. As a result, the GPA list may not contain CANCEL and REMIGRATE operations.

In case of an import error, then in addition to aborting the import of the current TD, an abort of an S4 resumption session also generates a new S4 anti-replay nonce, preventing a new S4 resumption session from starting.

Interruptibility: TDH.IMPORT.MEM is interruptible. If a pending interrupt is detected during operation, TDH.IMPORT.MEM returns with a TDX_INTERRUPTED_RSUMABLE status in RAX. RCX is updated with the next list entry index to process, so the host VMM may re-invoke TDH.IMPORT.MEM immediately after handling the interrupt, keeping the same inputs except setting R10.RESUME to 1.

Cache Lines Flushing (Future): The following applies to future platforms, if cache line flushing is required, as enumerated by TDX_FEATURES0.CLFLUSH_BEFORE_ALLOC (bit 23).

For each page that is imported for the first time (i.e., MIGRATE operation) not in-place, the host VMM should ensure that no cache lines associated with the separately provided physical page that is to be converted to a new TD private page are in a Modified state, as described in the [Base Spec].

This is not required for pages that are imported in-place. It is also not required for re-import (i.e., REMIGRATE operation).

Removed Page Initialization: On platforms which do not use ACT, after any private pages have been removed by a CANCEL operation, the host VMM should initialize their content before they are reused as non-private pages, as described in the [Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.60: TDH.IMPORT.MEM Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[5]
Explicit	RCX	HPA	GPA List page	GPA_LIST	RW	Shared	4KB	None	None	None	None
Explicit	RDX	HPA	TDR page	TDR	RW	Opaque	4KB	Shared	Shared	Shared	None
Explicit	R8	HPA	MBMD	MBMD	R	Shared	128B	None	None	None	None
Explicit	R9	HPA	Migration buffer list	PAGE_LIST	R	Shared	4KB	None	None	None	None
Explicit	R10	Index	Mig. Stream context	Mig. Stream context	RW	Opaque	N/A	Exclusive	N/A	N/A	N/A
Explicit	R11	HPA	MAC list page 1	MAC list	R	Shared	4KB	None	None	None	None
Explicit	R12	HPA	MAC list page 2	MAC list	R	Shared	4KB	None	None	None	None
Explicit	R13	HPA	Destination page list	Blob	RW	Private	4KB	Exclusive	Shared	Shared	None
Explicit	R14	HPA	L2 attributes list page	L2 page attributes	R	Shared	4KB	None	None	None	None
Explicit	N/A	GPA	TD private pages (via GPA list)	Blob	None	Private	4KB	None	None	None	None
Explicit	N/A	HPA	Migration buffer pages (via page list)	Blob	RW	Shared	4KB	None	None	None	None
Explicit	N/A	HPA	Destination pages (via page list)	Blob	RW	Private	4KB	Exclusive	Shared	Shared	Exclusive
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A	N/A

[5] ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[5]
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	Shared	N/A	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive	N/A	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT trees	N/A	RW	Private	N/A	Shared(i)	N/A	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A	N/A

TDH.IMPORT.MEM checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. An import session is in progress.
6. The migration stream index is lower than TDCS.NUM_MIGS.

If successful, the function does the following:

7. If the RESUME input flag is 0, indicating that this is a new (not resumed) invocation of TDH.IMPORT.MEM:
   7.1. Initialize the migration stream if not done so far.
   7.2. Copy the MBMD into a temporary buffer.
   7.3. Check the MBMD fields.

   If passed:

   7.4. Build the 96b IV for this migration bundle by concatenating 0 as the direction bit, the stream index and MBMD's MB_COUNTER.
   7.5. Check the MAC based on the MAC'ed fields of MBMD.
8. Else (this is a resumption of a previously interrupted TDH.IMPORT.MEM):
   8.1. Check that the stream context's INTERRUPTED_FUNC contains TDH.IMPORT.MEM's leaf number.
   8.2. Check that the current inputs are the same as saved in the stream context when the function was interrupted.

If passed, process the GPA list:

Note: Error conditions that impact a single GPA list entry, but do not cause an import session about, do not cause an abort of TDH.IMPORT.MEM. Instead, the GPA list entry is updates with a proper status code.

9. For each entry in the GPA list, starting with RCX.FIRST_ENTRY and ending with RCX.LAST_ENTRY:
   9.1. Increment the migration stream context's IV_COUNTER
   9.2. Build the 96b IV for this migration bundle by concatenating the stream index and the stream context's IV_COUNTER.
   9.3. Accumulate page MAC based on the GPA list entry.
   9.4. If a page L2 attributes list was provided, accumulate page MAC based on the page L2 attributes list entry.
   9.5. If no operation is requested:
      9.5.1. Check that the calculated MAC value is equal to the provided page MAC value.

If passed:

9.5.2. Mark the corresponding new page list entry (if available) as invalid and continue to the next GPA list entry.

9.6. Walk the SEPT based on the GPA and level operands and find the leaf entry for the page.

9.7. Check that the SEPT entry state is allowed for page import.

9.8. If import is in the out-of-order phase, check that the requested operation in first-time import.

9.9. If the requested operation if import or re-import, and the page state is not PENDING, check that a migration buffer is provided, and its address is a valid shared address.

9.10. If the requested operation is first-time migrate:

9.10.1. Check that the SEPT entry state is either FREE or REMOVED.

9.10.2. If the SEPT entry state is REMOVED, check that the has not been removed in the current migration epoch.

If passed:

9.10.3. If no new page list entry is provided, and a migration buffer is provided, this indicates in-place import. If the page is not PENDING, copy the migration buffer content to a temporary buffer. The migration buffer page will become the new TD private page.

9.10.4. Else, check that the new page list entry is a valid shared HPA.

9.10.5. If the page is not PENDING, decrypt the migration buffer or temporary buffer into the new TD page. Use direct writes (MOVDIR64B) and accumulate MAC.

9.10.6. Check that the calculated MAC value is equal to the provided page MAC value.

If passed:

9.10.7. On platforms which use ACT, update the new TD page ACT bit to private.

9.10.8. Update the new TD page PAMT entry; record the current migration epoch value in PAMT.BEPOCH.

9.10.9. Update the SEPT entry.

9.10.10. If a page L2 attributes list was provided, then for each valid L2 attributes entry in the in the page attributes list entry:

9.10.10.1. Check that the alias VM index is not higher than TDCS.NUM_L2VMS

9.10.10.2. Walk the L2 SEPT based on the GPA and level operands and find the FREE entry for the page alias.

9.10.10.3. Update the L2 SEPT entry based on the page L2 attributes list entry and the new TD page HPA.

9.11. Else, if the requested operation is re-migrate:

9.11.1. Check that the SEPT entry state is either MAPPED or PENDING.

9.11.2. Using the page's PAMT.BEPOCH, check that the page has not been imported in the current migration epoch.

If passed:

9.11.3. Record the current migration epoch value in PAMT.BEPOCH.

9.11.4. If the page is not PENDING, decrypt the migration buffer or temporary buffer into the new TD page. Use direct writes (MOVDIR64B) and accumulate MAC.

9.11.5. Check that the calculated MAC value is equal to the provided page MAC value.

If passed:

9.11.6. For each existing L2 page mapping or, if a page attributes list was provided, each valid L2 page attributes entry in the page attributes list entry:

9.11.6.1. In the page attributes list entry (if provided), check that the alias VM index is not higher than TDCS.NUM_L2VMS

9.11.6.2. Walk the L2 SEPT based on the GPA and level operands and find the leaf entry for the L2 page.

9.11.6.3. Update the L2 SEPT entry based on the page attributes list entry (if provided). If there is an existing L2 page mapping and no new L2 page attributes were provided, set the L2 SEPT entry state to L2_FREE.

9.11.7. Update the SEPT entry.

9.12. Else, if the requested operation is migration cancel:

9.12.1. Check that the SEPT entry state indicates that the page has been exported.

9.12.2. Calculate MAC over the GPA list entry and check that the value is equal to the provided page MAC value.

9.12.3. Using the page's PAMT.BEPOCH, check that the page has not been imported in the current migration epoch.

If passed:

9.12.4. For each existing L2 mapping of the page:
9.12.4.1. Walk the L2 SEPT based on the GPA and level operands and find the leaf entry for the page.
9.12.4.2. Set the L2 SEPT entry state to L2_FREE.
9.12.5. Update the SEPT entry; set the state to REMOVED and record the current migration epoch in the HPA.
9.12.6. On platforms which use ACT, overwrite the canceled TD page content with the TD's random overwrite value, using MOVDIR64B, and clear the corresponding ACT bit.
9.12.7. Update the PAMT entry of the canceled page to PT_NDA.
9.13. If this is not the last round and there is a pending interrupt:
9.13.1. Save intermediate state in the migration stream context.
9.13.2. Terminate TDH.EXPORT.MEM with a TDX_INTERRUPTED_RESUMABLE status.
9.14. Else, advance to the next entry in the GPA list, if applicable.
10. Once the GPA list has been fully processed, update the migration stream expected MB counter field.

Completion Status Codes

Table 5.61: TDH.IMPORT.MEM Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_DISALLOWED_IMPORT_OVER_REMOVED	
TDX_EPT_ENTRY_STATE_INCORRECT	
TDX_EPT_WALK_FAILED	
TDX_IMPORT_MISMATCH	
TDX_INTERRUPTED_RESUMABLE	
TDX_INVALID_MBMD	
TDX_INVALID_PAGE_MAC	
TDX_INVALID_RESUMPTION	
TDX_MIGRATED_IN_CURRENT_EPOCH	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	Operation is successful. **Note:** Processing of some GPA list entries may have encountered errors, but this did not cause an abort of the overall operation. The number such errors is reported in the lower 32 bits of the completion status.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.16. TDH.IMPORT.STATE.IMMUTABLE Leaf

TDH.IMPORT.STATE.IMMUTABLE starts a new import session and exports the TD's immutable state as a multi-page migration bundle.

TDH.IMPORT.STATE.IMMUTABLE is also used for starting a new S4 resumption session.

Table 5.62: TDH.IMPORT.STATE.IMMUTABLE Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	SEAMCALL instruction leaf number and version, see 5.4.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the SEAMCALL interface function
		23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
		63:24	Reserved	Must be 0
RCX	TDR	Destination TD handle and flags		
		Bits	**Name**	**Description**
		0	Import Type	0: TD Import 1: S4 Resumption
		11:1	Reserved	Must be 0
		51:12	TDR HPA	HPA[51:12] of the destination TD's TDR page (HKID bits must be 0)
		63:52	Reserved	Must be 0
R8	MBMD	HPA and size of memory of an MBMD structure in memory:		
		Bits	**Name**	**Description**
		51:0	HPA	Bits 51:0 of the host physical address (including HKID bits)
		63:52	Size	Size of the memory buffer containing MBMD, in bytes
R9	PAGE_LIST_INFO	Migration buffers list information – see 3.12.6.1		
R10	MIG_STREAM	Migration stream and resume flag:		
		Bits	**Name**	**Description**
		15:0	MIGS_INDEX	Migration stream index – must be 0
		62:16	RESERVED	Reserved: must be 0
		63	RESUME	0: This is a new invocation 1: This is resumption of a previously interrupted operation

Table 5.63: TDH.IMPORT.STATE.IMMUTABLE Output Operands Definition

Operand	Name	Description
RAX	STATUS	SEAMCALL instruction return code, see 5.4.1
RCX	EXTENDED_ ERROR_INFO1	In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RCX is unmodified. In case of an error related to non-memory state field import, as indicated by RAX, RCX contains the offending field identifier. See the status codes table below. In other cases, RCX returns 0.
RDX	EXTENDED_ ERROR_INFO2	In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RDX is unmodified. In other cases, RDX returns 0.
AVX, AVX2 and AVX512 state		May be reset to the architectural RESET state
Other		Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.IMPORT.STATE.IMMUTABLE starts a new import session. It imports the TD's immutable state migration bundle previously exported by TDH.EXPORT.STATE.IMMUTABLE. The migration bundle includes an MBMD and a set of 4KB pages.

TDH.IMPORT.STATE.IMMUTABLE is also used for starting a new S4 resumption session.

TD immutable state is verified by TDH.IMPORT.STATE.IMMUTABLE against target platform capabilities and Intel TDX module version, capabilities and configuration. The checks are similar, but not identical, to the TD_PARAMS checks done on the source platform by TDH.MNG.INIT.

Enumeration: Availability of TDH.IMPORT.STATE.IMMUTABLE is enumerated by either TDX_FEATURES0.TD_MIGRATION (bit 0) or TDX_FEATURES0.S4 (bit 13), readable by TDH.SYS.RD* (see 3.3.3.1), being set to 1. If not supported, calling TDH.IMPORT.STATE.IMMUTABLE returns a TDX_OPERAND_INVALID(RAX) status.

Interruptibility: TDH.IMPORT.STATE.IMMUTABLE is interruptible. The host VMM is expected to invoke it in a loop until it returns with either a success indication or with a non-recoverable error indication.

Import Abort: A failed TDH.IMPORT.STATE.IMMUTABLE marks (except in cases where the imported TD state has not been modified) the target TD as IMPORT_FAILED; by design, it will not run. This is indicated by the FATAL bit (61) of the completion status returned in RAX.

S4 Resumption Abort: In addition to aborting the import of the current TD, an abort of an S4 resumption session also generates a new S4 anti-replay nonce, preventing a new S4 resumption session from starting.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.64: TDH.IMPORT.STATE.IMMUTABLE Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	RW	Opaque	4KB	Exclusive	Shared	Shared

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	R8	HPA	MBMD	MBMD	R	Shared	128B	None	None	None
Explicit	R9	HPA	Page list	PAGE_LIST	R	Shared	4KB	None	None	None
Explicit	R10	Index	Mig. Stream context	Mig. Stream context	RW	Opaque	N/A	Exclusive	N/A	N/A
Explicit	N/A	HPA	Source pages (via page list)	Blob	R	Shared	4KB	None	None	None
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	PL.S4_STATE	N/A	RW	Hidden	N/A	Exclusive	N/A	N/A

TDH.IMPORT.STATE.IMMUTABLE checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. The TD has not been initialized (TDCS.OP_STATE is UNINITIALIZED).
6. If the TDX module doesn't support S4: A Migration TD has been bound to the source TD, and no migration session is in progress: Migration Session State is MIG_TD_BOUND.
7. The migration stream index is 0.
8. The buffer provided for MBMD is large enough and fits within a 4KB page.

If successful, the function does the following:

9. If the RESUME input flag is 0, indicating this is a new invocation of TDH.IMPORT.STATE.IMMUTABLE (not a resumption of a previously interrupted one):
   - 9.1. If IMPORT_TYPE is 0 (TD import):
     - 9.1.1. If the TDX module supports S4: A Migration TD has been bound to the source TD, and no migration session is in progress: Migration Session State is MIG_TD_BOUND.
     - 9.1.2. Check that a valid migration decryption key has been set by the Migration TD. If this is not the first migration session, then the migration key must have been set after the previous migration session has started.

       Note: There is no explicit check that a migration TD is bound; this is implied by the above check.

   - 9.2. Else (IMPORT_TYPE is 1 (S4 resumption) – applicable only if the TDX module supports S4):
     - 9.2.1. Check that PL.S4_STATE is either S4_IDLE or S4_IMPORT.
     - 9.2.2. Create the S4 decryption key and set TDCS.MIG_DEC_WORKING_KEY to this value.
     - 9.2.3. Set TDCS.MIG_WORKING_VERSION to S4_MIG_VERSION (an TDX module constant).
     - 9.2.4. Set TDCS.S4_MIGRATED to TRUE.
     - 9.2.5. Atomically increment PL.S4_IMP_INDEX.

   If passed:

   - 9.3. Initialize the migration context in TDCS:
     - 9.3.1. Copy the migration keys to working migration keys that will be used throughout the import session.

9.3.2. Generate a new migration encryption key, to be used in the next migration session.

If passed:

9.3.3. Set all migration streams' INITIALIZED flag to 0 and ENABLED flags to 1.

9.4. Initialize the current migration stream.

9.5. Copy the MBMD into the migration context.

9.6. Check the MBMD fields.

If passed:

9.7. Build the 96b IV for this migration bundle by concatenating the stream index and the stream context's IV_COUNTER.

9.8. Accumulate MAC based on the MAC'ed fields of MBMD.

10. Else (this is a resumption of a previously interrupted TDH.IMPORT.STATE.IMMUTABLE):

10.1. Check that the resumption is valid:

10.1.1. The stream context indicates there's a valid interruption state.

10.1.2. The current SEAMCALL leaf number and the PAGE_OR_LIST operand are the same as in the interruption state.

10.2. Check that the migration stream is enabled.

10.3. Restore the previously saved page list index from the migration context.

If passed:

11. Repeat importing 4KB pages until all immutable state is imported or until a pending interrupt is detected:

11.1. Get the 4KB next page HPA from it from the page list.

11.2. Use the migration key and the migration stream context to decrypt the 4KB internal buffer into an internal temporary 4KB buffer and update the MAC calculation.

11.3. Parse the metadata list and write the control structure fields using the algorithm described in 5.3.2.3. Check each TDR or TDCS field for compatibility.

If passed:

11.4. If all metadata lists have been imported:

11.4.1. Check that the accumulated MAC value is equal to the saved MBMD's MAC value.

11.4.2. Check that all global, TDR and TDCS metadata fields required to be imported by TDH.IMPORT.STATE.IMMUTABLE have indeed been imported.

11.4.3. Do validity checks of TDR and TDCS metadata fields that can only be checked at this stage.

11.4.4. Initialize TDR and TDCS fields that need to be initialized at the beginning of the import session.

11.4.5. Mark the migration stream context's interrupted state as invalid.

11.4.6. Increment the migration stream context's EXPECTED_MB_COUNTER.

11.4.7. Set TDCS.TOTAL_MB to 1.

11.4.8. Set TDCS.OP_STATE to MEMORY_IMPORT.

11.4.9. Terminate TDH.IMPORT.STATE.IMMUTABLE with a TDX_SUCCESS status.

11.5. Else, if there is a pending interrupt:

11.5.1. Save the interruption state to the stream context

11.5.2. Terminate TDH.IMPORT.STATE.IMMUTABLE with a TDX_INTERRUPTED_RESUMABLE status.

Completion Status Codes

Table 5.65: TDH.IMPORT.STATE.IMMUTABLE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INCORRECT_MBMD_MAC	
TDX_INTERRUPTED_RESUMABLE	
TDX_INVALID_MBMD	
TDX_INVALID_METADATA_LIST_HEADER	
TDX_INVALID_RESUMPTION	
TDX_METADATA_FIELD_ID_INCORRECT	Field ID or sequence header is returned in RCX

Completion Status Code	Description
TDX_METADATA_FIELD_NOT_WRITABLE	Field ID is returned in RCX
TDX_METADATA_FIELD_VALUE_NOT_VALID	Field ID is returned in RCX
TDX_METADATA_LIST_OVERFLOW	
TDX_MIGRATION_SESSION_KEY_NOT_SET	
TDX_MIGRATION_STREAM_STATE_INCORRECT	
TDX_NUM_MIGS_HIGHER_THAN_CREATED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_REQUIRED_METADATA_FIELD_MISSING	Required field ID is returned in RCX
TDX_RND_NO_ENTROPY	Failed to generate a random migration encryption key. This is typically caused by an entropy error of the CPU's random number generator, and may be impacted by RDSEED, RDRAND or PCONFIG executing on other LPs. The operation should be retried.
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	
TDX_VIRTUAL_MSR_VALUE_NOT_VALID	

5.4.17. TDH.IMPORT.STATE.TD Leaf

TDH.IMPORT.STATE.TD imports the TD-scope mutable state as a multi-page migration bundle.

Table 5.66: TDH.IMPORT.STATE.TD Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	SEAMCALL instruction leaf number and version, see 5.4.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the SEAMCALL interface function
		23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
		63:24	Reserved	Must be 0
RCX	TDR	HPA of Destination TD TDR page (HKID bits must be 0)		
R8	MBMD	HPA and size of memory of an MBMD structure in memory:		
		Bits	**Name**	**Description**
		51:0	HPA	Bits 51:0 of the host physical address (including HKID bits)
		63:52	Size	Size of the memory buffer containing MBMD, in bytes
R9	PAGE_LIST_INFO	Migration buffers list information – see 3.12.6.1		
R10	MIG_STREAM	Migration stream and resume flag:		
		Bits	**Name**	**Description**
		15:0	MIGS_INDEX	Migration stream index – must be 0
		62:16	RESERVED	Reserved: must be 0
		63	RESUME	0: This is a new invocation 1: This is resumption of a previously interrupted operation

Table 5.67: TDH.IMPORT.STATE.TD Output Operands Definition

Operand	Name	Description
RAX	STATUS	SEAMCALL instruction return code, see 5.4.1
RCX	EXTENDED_ ERROR_INFO1	In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RCX is unmodified. In case of an error related to non-memory state field import, as indicated by RAX, RCX contains the offending field identifier. See the status codes table below. In other cases, RCX returns 0.
RDX	EXTENDED_ ERROR_INFO2	In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RDX is unmodified. In other cases, RDX returns 0.

Operand	Name	Description
AVX, AVX2 and AVX512 state		May be reset to the architectural RESET state
Other		Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.IMPORT.STATE.TD imports the TD-scope mutable state migration bundle previously exported by TDH.EXPORT.STATE.TD. The migration bundle includes an MBMD and a set of 4KB pages.

TD-scope mutable state is verified by TDH.IMPORT.STATE.TD against target platform capabilities and Intel TDX module version, capabilities and configuration.

Enumeration: Availability of TDH.IMPORT.STATE.TD is enumerated by either TDX_FEATURES0.TD_MIGRATION (bit 0) or TDX_FEATURES0.S4 (bit 13), readable by TDH.SYS.RD* (see 3.3.3.1), being set to 1. If not supported, calling TDH.IMPORT.STATE.TD returns a TDX_OPERAND_INVALID(RAX) status.

Interruptibility: TDH.IMPORT.STATE.TD is interruptible. The host VMM is expected to invoke it in a loop until it returns with either a success indication or with a non-recoverable error indication.

Import Abort: A failed TDH.IMPORT.STATE.TD marks (except in cases where the imported TD state has not been modified) the target TD as IMPORT_FAILED; by design, it will not run. This is indicated by the FATAL bit (61) of the completion status returned in RAX.

S4 Resumption Abort: In addition to aborting the import of the current TD, an abort of an S4 resumption session also generates a new S4 anti-replay nonce, preventing a new S4 resumption session from starting.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.68: TDH.IMPORT.STATE.VP Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	R	Opaque	4KB	Exclusive	Shared	Shared
Explicit	R8	HPA	MBMD	MBMD	R	Shared	128B	None	None	None
Explicit	R9	HPA	Page list	PAGE_LIST	R	Shared	4KB	None	None	None
Explicit	R10	Index	Mig. Stream context	Mig. Stream context	RW	Opaque	N/A	Exclusive	N/A	N/A
Explicit	N/A	HPA	Source pages (via page list)	Blob	R	Shared	4KB	None	None	None
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	Exclusive(i)	N/A	N/A

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	N/A	Service TD bindings table	N/A	R	Hidden	N/A	Exclusive(i)	N/A	N/A

TDH.IMPORT.STATE.TD checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. An import session is in progress, but TD-scope mutable state has not been imported yet (TDCS.OP_STATE is MEMORY_IMPORT).
6. The migration stream index is 0.
7. The migration stream is enabled.
8. The buffer provided for MBMD is large enough and fits within a 4KB page.

If successful, the function does the following:

9. If the RESUME input flag is 0, indicating this is a new invocation of TDH.IMPORT.STATE.TD (not a resumption of a previously interrupted one):
   - 9.1. Copy the MBMD into the migration context.
   - 9.2. Check the MBMD fields.

   If passed:

   - 9.3. Build the 96b IV for this migration bundle by concatenating the stream index and the stream context's IV_COUNTER.
   - 9.4. Accumulate MAC based on the MAC'ed fields of MBMD.
10. Else (this is a resumption of a previously interrupted TDH.IMPORT.STATE.IMMUTABLE):
    - 10.1. Check that the resumption is valid:
      - 10.1.1. The stream context indicates there's a valid interruption state.
      - 10.1.2. The current SEAMCALL leaf number and the PAGE_OR_LIST operand are the same as in the interruption state.
    - 10.2. Restore the previously saved page list index from the migration context.

If passed:

11. Repeat importing 4KB pages until all immutable state is imported or until a pending interrupt is detected:
    - 11.1. Get the 4KB next page HPA from it from the page list.
    - 11.2. Use the migration key and the migration stream context to decrypt the 4KB internal buffer into an internal temporary 4KB buffer and update the MAC calculation.
    - 11.3. Parse the metadata list and write the control structure fields using the algorithm described in 5.3.2.3. Check each TDR or TDCS field for compatibility.

    If passed:

    - 11.4. If all metadata lists have been imported:
      - 11.4.1. Check that the accumulated MAC value is equal to the saved MBMD's MAC value.
      - 11.4.2. Check that all TDR and TDCS fields required to be imported by TDH.IMPORT.STATE.TD have indeed been imported.
      - 11.4.3. Initialize TDR and TDCS fields that need to be initialized at the end of the import session.
      - 11.4.4. Mark the migration stream context's interrupted state as invalid.
      - 11.4.5. Increment the migration stream context's EXPECTED_MB_COUNTER.
      - 11.4.6. Increment TDCS.TOTAL_MB.
      - 11.4.7. Set TDCS.OP_STATE to STATE_IMPORT.
      - 11.4.8. Terminate TDH.IMPORT.STATE.TD with a TDX_SUCCESS status.

11.5. Else, if there is a pending interrupt:
  11.5.1. Save the interruption state to the stream context
  11.5.2. Terminate TDH.IMPORT.STATE.TD with a TDX_INTERRUPTED_RESUMABLE status.

Completion Status Codes

Table 5.69: TDH.IMPORT.STATE.TD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	
TDX_INCORRECT_MBMD_MAC	
TDX_INTERRUPTED_RESUMABLE	
TDX_INVALID_MBMD	
TDX_INVALID_METADATA_LIST_HEADER	
TDX_INVALID_RESUMPTION	
TDX_METADATA_FIELD_ID_INCORRECT	Field ID or sequence header is returned in RCX
TDX_METADATA_FIELD_NOT_WRITABLE	Field ID is returned in RCX
TDX_METADATA_FIELD_VALUE_NOT_VALID	Field ID is returned in RCX
TDX_METADATA_LIST_OVERFLOW	
TDX_MIGRATION_STREAM_STATE_INCORRECT	
TDX_REQUIRED_METADATA_FIELD_MISSING	Required field ID is returned in RCX
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.18. TDH.IMPORT.STATE.VP Leaf

TDH.IMPORT.STATE.VP imports the VCPU-scope mutable state as a multi-page migration bundle.

Table 5.70: TDH.IMPORT.STATE.VP Input Operands Definition

Operand		Description		
RAX	Leaf and Version	SEAMCALL instruction leaf number and version, see 5.4.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the SEAMCALL interface function
		23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
		63:24	Reserved	Must be 0
RCX	TDVPR	HPA of the destination TD VCPU's TDVPR page (HKID bits must be 0)		
R8	MBMD	HPA and size of memory of an MBMD structure in memory:		
		Bits	**Name**	**Description**
		51:0	HPA	Bits 51:0 of the host physical address (including HKID bits)
		63:52	Size	Size of the memory buffer containing MBMD, in bytes
R9	PAGE_LIST_INFO	Migration buffers list information – see 3.12.6.1		
R10	MIG_STREAM	Migration stream and resume flag:		
		Bits	**Name**	**Description**
		15:0	MIGS_INDEX	Migration stream index If N migration streams have been created by TDH.MIG.STREAM.CREATE, then MIGS_INDEX must be lower than N-1.
		62:16	RESERVED	Reserved: must be 0
		63	RESUME	0: This is a new invocation 1: This is resumption of a previously interrupted operation

Table 5.71: TDH.IMPORT.STATE.VP Output Operands Definition

Operand	Name	Description
RAX	STATUS	SEAMCALL instruction return code, see 5.4.1
RCX	EXTENDED_ ERROR_INFO1	In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RCX is unmodified. In case of an error related to non-memory state field import, as indicated by RAX, RCX contains the offending field identifier. See the status codes table below. In other cases, RCX returns 0.
RDX	EXTENDED_ ERROR_INFO2	In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RDX is unmodified. In other cases, RDX returns 0.

Operand	Name	Description
AVX, AVX2 and AVX512 state		May be reset to the architectural RESET state
Other		Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.IMPORT.STATE.VP imports the VCPU-scope mutable state migration bundle previously exported by TDH.EXPORT.STATE.VP. The migration bundle includes an MBMD and a set of 4KB pages.

VCPU-scope mutable state is verified by TDH.IMPORT.STATE.VP against target platform capabilities and Intel TDX module version, capabilities and configuration.

Enumeration: Availability of TDH.IMPORT.STATE.VP is enumerated by either TDX_FEATURES0.TD_MIGRATION (bit 0) or TDX_FEATURES0.S4 (bit 13), readable by TDH.SYS.RD* (see 3.3.3.1), being set to 1. If not supported, calling TDH.IMPORT.STATE.VP returns a TDX_OPERAND_INVALID(RAX) status.

Interruptibility: TDH.IMPORT.STATE.VP is interruptible. The host VMM is expected to invoke it in a loop until it returns with either a success indication or with a non-recoverable error indication.

Import Abort: A failed TDH.IMPORT.STATE.VP marks (except in cases where the imported TD state has not been modified) the target TD as IMPORT_FAILED; by design, it will not run. This is indicated by the FATAL bit (61) of the completion status returned in RAX.

S4 Resumption Abort: In addition to aborting the import of the current TD, an abort of an S4 resumption session also generates a new S4 anti-replay nonce, preventing a new S4 resumption session from starting.

VCPU Association: TDH.IMPORT.VP associates the TD VCPU with the current LP. This requires that the VCPU will not be associated with another LP – for details, see the [TDX Module Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.72: TDH.IMPORT.STATE.VP Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDVPR page	TDVPS	RW	Opaque	4KB	Exclusive	Shared	Shared
Explicit	R8	HPA	MBMD	MBMD	R	Shared	128B	None	None	None
Explicit	R9	HPA	Page list	PAGE_LIST	R	Shared	4KB	None	None	None
Explicit	R10	Index	Mig. Stream context	Mig. Stream context	RW	Opaque	N/A	Exclusive	N/A	N/A
Explicit	N/A	HPA	Source pages (via page list)	Blob	R	Shared	4KB	None	None	None
Implicit	N/A	HPA	TDR page	TDR	R	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	None	N/A	N/A

TDH.IMPORT.STATE.VP checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDVPR page metadata in PAMT must be correct (PT must be PT_TDVPR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. An import session is in progress and TD-scope mutable state has been imported (TDCS.OP_STATE is STATE_IMPORT).
6. The migration stream index is lower than TDCS.NUM_MIGS.
7. The number of pages allocated to this TDVPS is correct.
8. The VCPU has not been initialized yet (TDVPS.VCPU_STATE is VCPU_UNINITIALIZED).
9. The buffer provided for MBMD is large enough.

If successful, the function does the following:

10. Associate the VCPU with the current LP, and update TD VMCS using the algorithm described in 5.3.1.

If passed:

11. If the RESUME input flag is 0, indicating this is a new invocation of a previously interrupted TDH.IMPORT.STATE.VP (not a resumption of a previously interrupted one):
    11.1. Copy the MBMD into the migration context.
    11.2. Check the MBMD fields.

    If passed:

    11.3. Build the 96b IV for this migration bundle by concatenating the stream index and the stream context's IV_COUNTER.
    11.4. Accumulate MAC based on the MAC'ed fields of MBMD.
    11.5. Atomically increment the TD's migrated VCPU counter (TDCS.NUM_MIGRATED_VCPUS), and check that number of VCPUs (TDCS.NUM_VCPUS) has not been exceeded.
12. Else (this is a resumption of a previously interrupted TDH.IMPORT.STATE.VP):
    12.1. Check that the resumption is valid:
        12.1.1. The stream context indicates there's a valid interruption state.
        12.1.2. The current SEAMCALL leaf number and the PAGE_OR_LIST operand are the same as in the interruption state.

If passed:

13. Repeat importing 4KB pages until all TD-scope state is imported or until a pending interrupt is detected:
    13.1. Get the 4KB next page HPA from it from the page list.
    13.2. Use the migration key and the migration stream context to decrypt the 4KB internal buffer into an internal temporary 4KB buffer and update the MAC calculation.
    13.3. Parse the metadata list and write the control structure fields using the algorithm described in 5.3.2.3. Check each TDVPS field for compatibility.

    If passed:

    13.4. If all metadata lists have been imported:
        13.4.1. Check that the accumulated MAC value is equal to the saved MBMD's MAC value.
        13.4.2. Check that all TDVPS fields required to be imported by TDH.IMPORT.STATE.VP have indeed been imported.
        13.4.3. Check and initialize TDVPS fields that need to be initialized at the end of the import session.
            13.4.3.1. If topology virtualization has been configured (as indicated by TDCS.TOPOLOGY_ENUM_CONFIGURED), check that the current VCPU's x2APIC_ID is unique.

If passed:

13.4.4. Mark the migration stream context's interrupted state as invalid.
13.4.5. Increment the migration stream context's EXPECTED_MB_COUNTER.
13.4.6. Increment TDCS.TOTAL_MB.
13.4.7. Terminate TDH.IMPORT.STATE.VP with a TDX_SUCCESS status.

13.5. Else, if there is a pending interrupt:
13.5.1. Save the interruption state to the stream context
13.5.2. Terminate TDH.IMPORT.STATE.VP with a TDX_INTERRUPTED_RESUMABLE status.

Completion Status Codes

Table 5.73: TDH.IMPORT.STATE.VP Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_ALL_VCPUS_IMPORTED	
TDX_INTERRUPTED_RESUMABLE	
TDX_INTERRUPTED_RESUMABLE	
TDX_INCORRECT_MBMD_MAC	
TDX_INTERRUPTED_RESUMABLE	
TDX_INVALID_MBMD	
TDX_INVALID_METADATA_LIST_HEADER	
TDX_INVALID_RESUMPTION	
TDX_METADATA_FIELD_ID_INCORRECT	Field ID or sequence header is returned in RCX
TDX_METADATA_FIELD_NOT_WRITABLE	Field ID is returned in RCX
TDX_METADATA_FIELD_VALUE_NOT_VALID	Field ID is returned in RCX
TDX_METADATA_LIST_OVERFLOW	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_REQUIRED_METADATA_FIELD_MISSING	Required field ID is returned in RCX
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	
TDX_VCPU_STATE_INCORRECT	
TDX_X2APIC_ID_NOT_UNIQUE	

5.4.19. TDH.IMPORT.TRACK Leaf

TDH.IMPORT.TRACK consumes an epoch token received from the source platform. It ends the current in-order import phase epoch and either starts a new epoch or starts the out-of-order import phase.

Table 5.74: TDH.IMPORT.TRACK Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function Must be 0
	23:16	Version Number	Selects the SEAMCALL interface function version
	63:24	Reserved	Must be 0
RCX	HPA of the source TD's TDR page (HKID bits must be 0)		
R8	HPA and size of memory of an MBMD structure in memory:		
	Bits	**Name**	**Description**
	51:0	HPA	Bits 51:0 of the host physical address (including HKID bits)
	63:52	Size	Size of the memory buffer containing MBMD, in bytes
R10	Migration stream index – must be 0		

Table 5.75: TDH.IMPORT.TRACK Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 5.4.1
AVX, AVX2 and AVX512 state	May be reset to the architectural RESET state
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.IMPORT.TRACK parses an epoch token received from the source platform. It checks that the epoch number indicated by the token is correct, and that all migration bundles indicated by the token have been received.

If successful, it ends the current import epoch, and as indicated by the epoch token either starts a new epoch or starts the out-of-order import phase.

Enumeration: Availability of TDH.IMPORT.TRACK is enumerated by either TDX_FEATURES0.TD_MIGRATION (bit 0) or TDX_FEATURES0.S4 (bit 13), readable by TDH.SYS.RD* (see 3.3.3.1), being set to 1. If not supported, calling TDH.IMPORT.TRACK returns a TDX_OPERAND_INVALID(RAX) status.

Import Abort: A failure may mark the target TD as IMPORT_FAILED; by design, it will not run. This is indicated by the FATAL bit (61) of the completion status returned in RAX.

S4 Resumption: When called during an S4 resumption session, TDH.IMPORT.TRACK only supports transitioning to the out-of-order import phase, with a start token MBMD generated by TDH.EXPORT.TRACK with IN_ORDER_DONE specified.

If an error is encountered, then in addition to aborting the import of the current TD, an abort of an S4 resumption session also generates a new S4 anti-replay nonce, preventing a new S4 resumption session from starting.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.76: TDH.IMPORT.TRACK Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	R	Opaque	4KB	Exclusive	Shared	Shared
Explicit	R8	HPA	MBMD buffer	MBMD	R	Shared	128B	None	None	None
Explicit	R10	Index	Mig. Stream context	Mig. Stream context	RW	Opaque	N/A	Exclusive	N/A	N/A
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	Exclusive(i)	N/A	N/A

TDH.IMPORT.TRACK checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS is allocated (TDR.NUM_TDCX is the required number).
5. An import session is in the in-order phase: TDCS.OP_STATE is either MEMORY_IMPORT or STATE_IMPORT.
6. An import session is in progress and the TD-scope state has been imported: TDCS.OP_STATE is STATE_IMPORT.
7. The migration stream index is 0.
8. The migration stream is initialized.
9. The buffer provided for MBMD is large enough.

If successful, the function does the following:

10. Copy the MBMD into a temporary buffer.
11. Check the MBMD fields:
    11.1. Check that SIZE is large enough.
    11.2. Check that MB_TYPE indicates an epoch token.
    11.3. Check that MIGS_INDEX is 0.
    11.4. Check that the MB_COUNTER value is equal to the migration stream's EXPECTED_RX_COUNTER.
    11.5. Check that MIG_EPOCH is higher than TDCS.MIG_EPOCH.
    11.6. Check that TOTAL_MB is equal to TDCS.TOTAL_MB + 1.
    11.7. Check that reserved fields are 0.

If passed:

12. Build the 96b IV for this migration bundle by concatenating the stream index and the stream context's IV_COUNTER.
13. Accumulate MAC based on the MAC'ed fields of MBMD and check that the value is the same as the MBMD's MAC field's value.

If passed:

14. If the MIG_EPOCH value provided in the MBMD is 0xFFFFFFFF, indicating the start of out-of-order phase:
    14.1. Check that all VCPUs have been imported

    If passed:

    14.2. Start the out-of-order import phase: set TDCS.OP_STATE to POST_IMPORT.
15. Set the stream context's EXPECTED_MB_COUNTER to 1.
16. Increment TDCS.TOTAL_MB.
17. Set TDCS.MIG_EPOCH to the MIG_EPOCH value provided in the MBMD.

Completion Status Codes

Table 5.77: TDH.IMPORT.TRACK Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INCORRECT_MBMD_MAC	
TDX_INVALID_MBMD	
TDX_MIGRATION_STREAM_STATE_INCORRECT	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SOME_VCPUS_NOT_MIGRATED	
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.20. TDH.MEM.PAGE.ADD Leaf

Add a 4KB private page to a TD, mapped to the specified GPA, filled with the given page image and encrypted using the TD ephemeral key, and update the TD measurement with the page properties.

Table 5.78: TDH.MEM.PAGE.ADD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the EPT entry that will map the new page – see 3.6.1: must be 0
	11:3	Reserved	Reserved: must be 0
	51:12	GPA	Bits 51:12 of the guest physical address to be mapped for the new Secure EPT page
	63:52	Reserved	Reserved: must be 0
RDX	Host physical address of the parent TDR page (HKID bits must be 0)		
R8	Host physical address of the target page to be added to the TD (HKID bits must be 0)		
R9	Host physical address (including HKID bits) of the source page image		

Table 5.79: TDH.MEM.PAGE.ADD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page and may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. In other cases, RCX returns 0.
RDX	Extended error information part 2 In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.PAGE.ADD adds a 4KB private page to a TD and maps it to the provided GPA. It copies the provided source page image to specified physical page using the TD's ephemeral private key and updates the TD measurement with the page properties. TDH.MEM.PAGE.ADD is used during TD build before the TD is initialized.

In-Place Add: It is allowed to set the TD page HPA in R8 to the same address as the source page HPA in R9. In this case the source page is converted to be a TD private page.

Cache Lines Flushing (Future): On future platforms, if cache line flushing is required, as enumerated by TDX_FEATURES0.CLFLUSH_BEFORE_ALLOC (bit 23), then before calling TDH.MEM.PAGE.ADD, the host VMM should ensure that no cache lines associated with the added physical page are in a Modified state, as described in the [Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.80: TDH.MEM.PAGE.ADD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[6]
Explicit	RCX	GPA	TD private page (GPA)[7]	Blob	RW	Private	4KB	N/A	N/A	N/A	N/A
Explicit	RDX	HPA	TDR page	Blob	RW	Opaque	4KB	Exclusive	Shared	Shared	None
Explicit	R8	HPA	TD private page (HPA)[7]	Blob	RW	Private	4KB	Exclusive	Shared	Shared	Exclusive
Explicit	R9	HPA	Source page	Blob	R	Shared	4KB	None	None	None	None
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Exclusive(i)	N/A	N/A	None
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive(i)	N/A	N/A	None
Implicit	N/A	GPA	Secure EPT tree	N/A	RW	Private	N/A	Exclusive(i)	N/A	N/A	None
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A	None

TDH.MEM.PAGE.ADD checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must have been initialized but not finalized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is INITIALIZED).
5. The target page metadata in PAMT must be correct (PT must be PT_NDA).

If successful, the function does the following:

6. Walk the Secure EPT based on the GPA operand and find the leaf EPT entry for the 4KB page.

[6] ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

[7] RCX and R8 denote the same TD private page operand, using HPA and GPA respectively

If the Secure EPT entry is marked as FREE, the function does the following:

7. On platforms which use ACT:
   7.1. If source and destination are overlapping, then:
      7.1.1. Read 4KB of data from source image to a temporary buffer.
      7.1.2. Update the destination's page bit in ACT to private.
      7.1.3. Write 4KB of data form temporary buffer to destination page using direct write (MOVDIR64B).
   7.2. Else:
      7.2.1. Update the destination's page bit in ACT to private.
      7.2.2. Copy the source image to the target TD page using the TD's ephemeral private HKID, and direct write (MOVDIR64B).
8. On platforms which do not use ACT: Copy the source image to the target TD page using the TD's ephemeral private HKID, and direct write (MOVDIR64B).
9. Update the parent Secure EPT entry with the target page HPA and MAPPED state.
10. Extend TDCS.MRTD with the target page GPA. Extension is done using SHA384 with a 128B extension buffer composed as follows:
    - Bytes 0 through 11 contain the ASCII string "MEM.PAGE.ADD".
    - Bytes 16 through 23 contain the GPA (in little-endian format).
    - All the other bytes contain 0.
11. Increment TDR.CHLDCNT.
12. Update the PAMT entry with the PT_REG page type and the TDR physical address as the OWNER.

Completion Status Codes

Table 5.81: TDH.MEM.PAGE.ADD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_STATE_INCORRECT	
TDX_EPT_WALK_FAILED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MEM.PAGE.ADD is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.21. TDH.MEM.PAGE.AUG Leaf

Dynamically add a 4KB or a 2MB private page to an initialized TD, mapped to the specified GPAs.

Table 5.82: TDH.MEM.PAGE.AUG Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the EPT entry that will map the new page – see 3.6.1: must be 0 (4KB) or 1 (2MB)
	11:3	Reserved	Reserved: must be 0
	51:12	GPA	Bits 51:12 of the guest physical address to be mapped for the new Secure EPT page
	63:52	Reserved	Reserved: must be 0
RDX	Host physical address of the parent TDR page (HKID bits must be 0)		
R8	Host physical address of the target page to be added to the TD (HKID bits must be 0)		

Table 5.83: TDH.MEM.PAGE.AUG Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page and may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. In other cases, RCX returns 0.
RDX	Extended error information part 2 In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.PAGE.AUG adds a 4KB or a 2MB private page to a TD and maps it to the provided GPA. The new page is mapped in a pending state and can be accessed only by the guest TD after it accepts it using TDCALL(TDG.MEM.PAGE.ACCEPT). TDH.MEM.PAGE.AUG does not initialize the new page and does not update the TD measurement.

Cache Lines Flushing (Future): On future platforms, if cache line flushing is required, as enumerated by TDX_FEATURES0.CLFLUSH_BEFORE_ALLOC (bit 23), then before calling TDH.MEM.PAGE.AUG, the host VMM should ensure that no cache lines associated with the added physical page are in a Modified state, as described in the [Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.84: TDH.MEM.PAGE.AUG Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[8]
Explicit	RCX	GPA	TD private page (GPA)[9]	Blob	None	Private	$2^{12+9*Level}$ Bytes	N/A	N/A	N/A	N/A
Explicit	RDX	HPA	TDR page	TDR	RW	Opaque	4KB	Shared	Shared	Shared	None
Explicit	R8	HPA	TD private page (HPA)[9]	Blob	None	Private	$2^{12+9*Level}$ Bytes	Exclusive	Shared[10]	Shared	Exclusive
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A	N/A
Implicit	N/A	GPA	Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	N/A	N/A	N/A

TDH.MEM.PAGE.AUG checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must be in one of the following states:
   4.1. The TD has been initialized locally by TDH.MNG.INIT and no migration session is in progress
   4.2. An export session is in progress its live export phase; TDH.EXPORT.PAUSE has not been invoked yet.
   4.3. An import session is in its live import phase, initiated by TDH.IMPORT.COMMIT.
5. The target page metadata in PAMT must be correct (PT must be PT_NDA for the entire 4KB or 2MB range).

If successful, the function does the following:

6. Walk the Secure EPT based on the GPA operand and find the leaf EPT entry for the 4KB or 2MB page.

[8] ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

[9] RCX and R8 denote the same TD private page operand, using HPA and GPA respectively

[10] Applicable for 4KB pages only

If the Secure EPT entry is marked as FREE, the function does the following:

7. Update the parent Secure EPT entry with the target page HPA and PENDING state.
8. Atomically increment TDR.CHLDCNT by 1 (for a 4KB page) or by 512 (for a 2MB page).
9. On platforms which use ACT, update the ACT page bit(s) to private.
10. Update the PAMT entry with the PT_REG page type and the TDR physical address as the OWNER.

Completion Status Codes

Table 5.85: TDH.MEM.PAGE.AUG Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_STATE_INCORRECT	
TDX_EPT_WALK_FAILED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MEM.PAGE.AUG is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.22. TDH.MEM.PAGE.DEMOTE Leaf

Split a large private TD page (2MB or 1GB) into 512 small pages (4KB or 2MB, respectively).

Table 5.86: TDH.MEM.PAGE.DEMOTE Input Operands Definition

<table>
<tr><th>Operand</th><th colspan="4">Description</th></tr>
<tr><td rowspan="4">RAX</td><td colspan="4">SEAMCALL instruction leaf number and version, see 5.4.1</td></tr>
<tr><td>Bits</td><td>Field</td><td colspan="2">Description</td></tr>
<tr><td>15:0</td><td>Leaf Number</td><td colspan="2">Selects the SEAMCALL interface function</td></tr>
<tr><td>23:16</td><td>Version Number</td><td colspan="2">Selects the SEAMCALL interface function version
Must be 0</td></tr>
<tr><td></td><td>63:24</td><td>Reserved</td><td colspan="2">Must be 0</td></tr>
<tr><td rowspan="5">RCX</td><td colspan="4">EPT mapping information:</td></tr>
<tr><td>Bits</td><td>Name</td><td colspan="2">Description</td></tr>
<tr><td>2:0</td><td>Level</td><td colspan="2">Level of the Secure EPT entry that maps the large page to be split: either 1 (2MB) or 2 (1GB) – see 3.6.1</td></tr>
<tr><td>11:3</td><td>Reserved</td><td colspan="2">Reserved: must be 0</td></tr>
<tr><td>51:12</td><td>GPA</td><td colspan="2">Bits 51:12 of the guest physical address of the large page to be split
Depending on the level, the following least significant bits must be 0:
Level 1 (2MB): Bits 20:12
Level 2 (1GB): Bits 29:12</td></tr>
<tr><td></td><td>63:52</td><td>Reserved</td><td colspan="2">Reserved: must be 0</td></tr>
<tr><td rowspan="10">RDX</td><td colspan="4">TD handle and flags:</td></tr>
<tr><td>Bits</td><td>Name</td><td colspan="2">Description</td></tr>
<tr><td>0</td><td>L2_SEPT_ADD_MODE</td><td colspan="2">New L2 SEPT pages addition mode:</td></tr>
<tr><td></td><td></td><td>Value</td><td>Name</td><td>Description</td></tr>
<tr><td></td><td></td><td>0</td><td>DENSE</td><td>New L2 SEPT pages are added, if provided by R9, R10 or R11.</td></tr>
<tr><td></td><td></td><td>1</td><td>SPARSE</td><td>New L2 SEPT pages are added, if provided by R9, R10 or R11, but only for L2 VMs where the L1 VMM has created a page alias (using TDG.MEM.PAGE.ATTR.WR).</td></tr>
<tr><td></td><td></td><td colspan="3">In both cases, a new L2 SEPT must be provided for L2 VMs where a page alias exists.
This bit is ignored if the TD has no L2 VMs.</td></tr>
<tr><td></td><td>11:1</td><td>Reserved</td><td colspan="2">Reserved: must be 0</td></tr>
<tr><td></td><td>51:12</td><td>TDR_HPA</td><td colspan="2">Bits 51:12 of the host physical address of the parent TDR page (HKID bits must be 0)</td></tr>
<tr><td></td><td>63:52</td><td>Reserved</td><td colspan="2">Reserved: must be 0</td></tr>
</table>

Operand	Description
R8	Host physical address of the new L1 Secure EPT page to be added to the TD (HKID bits must be 0)
R9	If the number of L2 VMs is >= 1, R9 contains the host physical address of the new Secure EPT page to be added to L2 VM #1's SEPT tree (HKID bits must be 0). Else (the number of L2 VMs is 0), R9 is ignored. If the value of R9 is NULL_PA (-1), no new SEPT page is added to L2 VM #1's SEPT. If the demoted TD private page has an L2 page alias for L2 VM #1, this is an error. Else, bit 63 of R9 is ignored. If L2_SEPT_ADD_MODE is 1, the new SEPT page is only used if the demoted TD private page has an L2 page alias for L2 VM #1. Else, the new SEPT page is always used.
R10	If the number of L2 VMs is >= 2, R10 contains the host physical address of the new Secure EPT page to be added to L2 VM #2's SEPT tree (HKID bits must be 0). Else (the number of L2 VMs is 0 or 1), R10 is ignored. If the value of R10 is NULL_PA (-1), no new SEPT page is added to L2 VM #2's SEPT. If the demoted TD private page has an L2 page alias for L2 VM #2, this is an error. Else, bit 63 of R10 is ignored. If L2_SEPT_ADD_MODE is 1, the new SEPT page is only used if the demoted TD private page has an L2 page alias for L2 VM #2. Else, the new SEPT page is always used.
R11	If the number of L2 VMs is >= 3, R11 contains the host physical address of the new Secure EPT page to be added to L2 VM #3's SEPT tree (HKID bits must be 0). Else (the number of L2 VMs is 0, 1 or 2), R11 is ignored. If the value of R11 is NULL_PA (-1), no new SEPT page is added to L2 VM #3's SEPT. If the demoted TD private page has an L2 page alias for L2 VM #3, this is an error. Else, bit 63 of R11 is ignored. If L2_SEPT_ADD_MODE is 1, the new SEPT page is only used if the demoted TD private page has an L2 page alias for L2 VM #3. Else, the new SEPT page is always used.

Table 5.87: TDH.MEM.PAGE.DEMOTE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RCX	In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESTARTABLE, RCX is unmodified. Else, RCX returns extended error information part 1. In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page; it may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. In other cases, RCX returns 0.
RDX	In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESTARTABLE, RDX is unmodified. Else, RDX returns extended error information part 2. In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0.

Operand	Description
R9	If TDH.MEM.PAGE.DEMOTE terminated successfully, the number of L2 VMs is >= 1 and the page whose HPA was provided in R9 was not used as an SEPT page for any reason, R9 is updated with bit 63 set to 1. Else, R9 is unmodified.
R10	If TDH.MEM.PAGE.DEMOTE terminated successfully, the number of L2 VMs is >= 2 and the page whose HPA was provided in R10 was not used as an SEPT page for any reason, R10 is updated with bit 63 set to 1. Else, R10 is unmodified.
R11	If TDH.MEM.PAGE.DEMOTE terminated successfully, the number of L2 VMs is >= 3 and the page whose HPA was provided in R11 was not used as an SEPT page for any reason, R11 is updated with bit 63 set to 1. Else, R11 is unmodified.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.PAGE.DEMOTE splits a large TD private page (2MB or 1GB) into 512 small pages (4KB or 2MB, respectively) and adds a new Secure EPT page to map those small pages. If the large page is mapped in any L2 SEPTs, TDH.MEM.PAGE.DEMOTE splits those mapping and adds new L2 Secure EPT pages to map the demoted page.

Enumeration: TDH.MEM.PAGE.DEMOTE support of non-blocking mapping resize is enumerated by TDX_FEATURES0.NON_BLOCKING_RESIZE (bit 35), readable using TDH.SYS.RD*.

Blocking and TLB Tracking: If the TDX module supports non-blocking mapping resize, no blocking and tracking of the demoted page is required.

Else, if the TD may be running, the demoted page must be blocked and TLB tracked. Else (e.g., TDH.MR.FINALIZE has not yet been executed, or the TD has been paused for export), no blocking and tracking is required.

PT_TR: If the TDX module supports non-blocking mapping resize, then PT_TR pages (i.e., former SEPT pages converted by TDH.MEM.PAGE.PROMOTE) owned by the same TD can be provided as new SEPT pages. TDH.MEM.PAGE.DEMOTE checks those pages for TLB tracking.

L2 SEPT Population: TDH.MEM.PAGE.DEMOTE supports multiple host VMM policies for populating the L2 SEPT trees.

- Dense mode is when the host VMM maintains an L2 SEPT page for each L1 SEPT page. This mode is selected by setting L2_SEPT_ADD_MODE to 0.
- Sparse mode is when the host VMM only maintains an L2 SEPT page for a certain L1 SEPT page on demand, i.e., when there's a need to map a TD private page in an L2 VM's GPA space. This mode is selected by setting L2_SEPT_ADD_MODE to 0. The host VMM provides new SEPT pages, but they are only used is a page alias exists for the relevant L2 VM.

The host VMM may also choose to maintain L2 SEPT trees only for a subset of the L2 VMs (e.g., if a TD is created with a certain number of L2 VMs but not all of them are currently in use). The host VMM can do so by providing NULL_PA as the new SEPT page(s) HPA.

Interruptibility: TDH.MEM.PAGE.DEMOTE is interruptible but not resumable. If a pending interrupt is detected during operation, TDH.MEM.PAGE.DEMOTE returns with a TDX_INTERRUPTED_RESTARTABLE status in RAX. No demote operation is done and no output operands except RAX are modified.

In such case, TDH.MEM.PAGE.DEMOTE should be invoked in a loop until it terminates successfully. The host VMM should be designed to avoid cases where interrupt storms prevent successful completion of TDH.MEM.PAGE.DEMOTE.

Cache Lines Flushing (Future): On future platforms, if cache line flushing is required, as enumerated by TDX_FEATURES0.CLFLUSH_BEFORE_ALLOC (bit 23), then before calling TDH.MEM.PAGE.DEMOTE, the host VMM should ensure that no cache lines associated with the added SEPT physical pages are in a Modified state, as described in the [Base Spec].

The table below shows the values of the SEPT HPA arguments in R9 – R11 when no error occurs (RAX returns TDX_SUCCESS).

Table 5.88: Meaning of TDH.MEM.PAGE.DEMOTE's SEPT HPA Arguments on Input and Output (No Error)

Value	R9 – R11 on Input	R9 – R11 on Output
NULL_PA (-1)	No new SEPT page to add	Unmodified
Bits 62:0: Valid HPA, HKID bits are 0 Bit 63: 0	Bits 62:0: HPA of new SEPT page to add Bit 63: Ignored	TDH.MEM.PAGE.DEMOTE terminated successfully and the new SEPT page has been added
Bits 62:0: Valid HPA, HKID bits are 0 Bit 63: 1		TDH.MEM.PAGE.DEMOTE terminated successfully and the new SEPT page has not been added

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.89: TDH.MEM.PAGE.DEMOTE Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource Name	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[11]
Explicit	RCX	GPA and Level	TD private page to split	Blob	None	Private	$2^{12+9*level}$ bytes	Exclusive	None	None	None
Explicit	RDX	HPA	TDR page	TDR	RW	Opaque	4KB	Shared	Shared	Shared	None
Explicit	R8	HPA	New L1 Secure EPT page	SEPT_PAGE	RW	Private	4KB	Exclusive	Shared	Shared	Exclusive
Explicit	R9, R10, R11	HPA	New L2 Secure EPT pages	SEPT_PAGE	RW	Private	4KB	Exclusive	Shared	Shared	Exclusive
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A	None
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A	None
Implicit	N/A	GPA	L1 Secure EPT Tree	N/A	RW	Private	N/A	Shared	N/A	N/A	None
Implicit	N/A	GPA	L1 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	N/A	N/A	None
Implicit	N/A	GPA	L2 Secure EPT Trees	N/A	RW	Private	N/A	Shared(i)	N/A	N/A	None

[11] ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

Explicit/ Implicit	Reg.	Ref Type	Resource Name	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[11]
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A	None

TDH.MEM.PAGE.DEMOTE checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must have been initialized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is either INITIALIZED or RUNNING).
5. The specified page level is either 1 (2MB) or 2 (1GB). See 3.6.1 for a definition of EPT level.

If successful, the function does the following:

6. Walk the L1 Secure EPT based on the GPA operand and locate the large TD private page to be demoted.
7. If the TDX module does not support non-blocking mapping resize, as enumerated by TDX_FEATURES0.NON_BLOCKING_RESIZE (bit 35), and the TD may run (its OP_STATE in either RUNNING, LIVE_EXPORT or LIVE_IMPORT), check page block and tracking:
   7.1. Check the Secure EPT entry: It must be a leaf BLOCKED or PENDING_BLOCKED entry.
   7.2. Check that TLB tracking has been done, based on the large TD page's PAMT.BEPOCH.
8. Else (no blocking and tracking is required):
   8.1. Check the Secure EPT entry: It must be a leaf MAPPED, BLOCKED, PENDING or PENDING_BLOCKED entry.

If passed:

9. Check that the new L1 SEPT and L2 SEPT pages metadata in PAMT is correct:
   9.1. If the TDX module supports non-blocking mapping resize and PAMT.PT is PT_TR:
      9.1.1. Check that the PAMT.OWNER of the PT_TR page is the current TD.
      9.1.2. If the TD may run (its OP_STATE in either RUNNING, LIVE_EXPORT or LIVE_IMPORT), check page block and tracking.
   9.2. Else, check that PAMT.PT is PT_NDA.

If passed:

10. Split the large TD private page PAMT entry into 512 PAMT entries at the lower level:
    10.1. Set the parent PAMT_2M or PAMT_1G entry state to PT_NDA.
    10.2. Set the 512 child PAMT4K or PAMT_2M entries respectively to PT_REG.
11. On platforms which use ACT, update the ACT bit of the new SEPT page to private.
12. Initialize the new Secure EPT page's 512 entries to MAPPED (if the original page was MAPPED or BLOCKED) or PENDING (if the original page was PENDING or PENDING_BLOCKED) pointing to the 512 consecutive small pages above. Use the TD's ephemeral private HKID and direct write (MOVDIR64B).
13. Atomically set the original Secure EPT entry to NL_MAPPED non-leaf entry pointing to the new Secure EPT page.
14. For each L2 VM, if L2_SEPT_ADD_MODE is 0 or there is an L2 mapping of the page to be demoted:
    14.1. Walk the L2 Secure EPT based on the GPA operand and locate the L2 Secure EPT parent entry of the page to be demoted.

    Note: If there is an L2 mapping of the page, this walk should not fail. The L2 SEPT entry state is implicit: It must be a leaf, mapped (L2_MAPPED) or blocked (L2_BLOCKED) entry. Else, the walk may fail – for this reason the update is done below only if all SEPT walks succeeded.

If passed:

15. For each L2 VM:
    15.1. If there is an L2 mapping of the page to be demoted:
       15.1.1. On platforms which use ACT, update the ACT bit of the new L2 SEPT page to private.

15.1.2. Initialize the new L2 Secure EPT page's 512 entries to L2_MAPPED (if the original page was MAPPED or BLOCKED) or PENDING (if the original page was PENDING or PENDING_BLOCKED) pointing to the 512 consecutive small pages above. Use the TD's ephemeral private HKID and direct write (MOVDIR64B).

15.2. Else, if L2_SEPT_ADD_MODE is 0:

15.2.1. Initialize the new L2 Secure EPT page's 512 entries to L2_FREE. Use the TD's ephemeral private HKID and direct write (MOVDIR64B).

15.3. Atomically set the original L2 Secure EPT entry to NL_MAPPED non-leaf entry pointing to the new L2 Secure EPT page.

16. Atomically increment TDR.CHLDCNT by 1.

16.1. Note that CHLDCNT counts the number of 4KB pages. The change is due only to the addition of the new Secure EPT page.

17. Update the PAMT entry of the new Secure-EPT page with the PT_EPT page type and the TDR physical address as the OWNER.

18. For each L2 VM:

18.1. If there was an L2 mapping of the page and the new L2 SEPT page was used:

18.1.1. Update the PAMT entry of each new L2 SEPT page with the PT_EPT page type and the TDR physical address as the OWNER.

18.1.2. Atomically increment TDR.CHLDCNT by 1.

Else:

18.1.3. Set bit 63 of the applicable output GPR (R9, R10 or R11) to 1, indicating that this page was not used as an SEPT page.

Completion Status Codes

Table 5.90: TDH.MEM.PAGE.DEMOTE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_STATE_INCORRECT	
TDX_EPT_WALK_FAILED	
TDX_GPA_RANGE_NOT_BLOCKED	
TDX_INTERRUPTED_RESTARTABLE	TDH.MEM.PAGE.DEMOTE's operation has been interrupted by an external event; it may be restarted (from its beginning) by calling it again.
TDX_L2_SEPT_PAGE_NOT_PROVIDED	
TDX_L2_SEPT_WALK_FAILED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MEM.PAGE.DEMOTE is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	

Completion Status Code	Description
TDX_TDCS_NOT_ALLOCATED	
TDX_TLB_TRACKING_NOT_DONE	

5.4.23. TDH.MEM.PAGE.PROMOTE Leaf

Merge 512 consecutive small private TD pages (4KB or 2MB) into one large page (2MB or 1GB, respectively).

Table 5.91: TDH.MEM.PAGE.PROMOTE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Versions 0 and 1 are supported. See enumeration details below.
	63:24	Reserved	Must be 0
RCX	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the Secure EPT entry that will map the merged large page: either 1 (2MB) or 2 (1GB) (see 3.6.1)
	11:3	Reserved	Reserved: must be 0
	51:12	GPA	Bits 51:12 of the guest physical address of the merged large page Depending on the level, the following least significant bits must be 0: Level 1 (2MB): Bits 20:12 Level 2 (1GB): Bits 29:12
	63:52	Reserved	Reserved: must be 0
RDX	TD handle and flags:		
	Bits	**Name**	**Description**
	0	NO_TRACK	Large GPA range blocking and TLB tracking mode 0: The merged large GPA range is checked for blocking and TLB tracking, and SEPT pages are removed. 1: The merged large GPA range is not checked for blocking and TLB tracking, and SEPT pages are converted to PT_TR pages. **Enumeration:** Support of NO_TRACK value of 1 is enumerated by TDX_FEATURES0.NON_BLOCKING_RESIZE (bit 35), readable using TDH.SYS.RD*.
	11:1	Reserved	Reserved: must be 0
	51:12	TDR_HPA	Bits 51:12 of the host physical address of the parent TDR page (HKID bits must be 0)
	63:52	Reserved	Reserved: must be 0

Table 5.92: TDH.MEM.PAGE.PROMOTE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1

Operand	Description
RCX	If TDH.MEM.PAGE.PROMOTE succeeded, RCX returns the HPA of the removed SEPT page (HKID bits are set to 0). In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESTARTABLE, RCX is unmodified. Else, RCX returns extended error information part 1. In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page; it may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. In other cases, RCX returns 0.
RDX	In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESTARTABLE, RDX is unmodified. Else, RDX returns extended error information part 2. In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0.
R9	If TDH.MEM.PAGE.PROMOTE version is 1 or higher: • If L2 VM #1's L2 SEPT page has been removed, R9 returns the HPA of that SEPT page (HKID bits are set to 0). • Else, R9 returns NULL_PA (-1). Else, R9 is unmodified.
R10	If TDH.MEM.PAGE.PROMOTE version is 1 or higher: • If L2 VM #2's L2 SEPT page has been removed, R10 returns the HPA of that SEPT page (HKID bits are set to 0). • Else, R10 returns NULL_PA (-1). Else, R10 is unmodified.
R11	If TDH.MEM.PAGE.PROMOTE version is 1 or higher: • If L2 VM #3's L2 SEPT page has been removed, R11 returns the HPA of that SEPT page (HKID bits are set to 0). • Else, R11 returns NULL_PA (-1). Else, R11 is unmodified.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.PAGE.PROMOTE merges 512 private pages, which are consecutive both in the HPA space and in the GPA space. The L1 SEPT page and all existing L2 SEPT pages at the requested GPA and level are removed.

All merged private pages must have the same Secure EPT leaf entry attributes and state, which must be either MAPPED or PENDING. All merged private pages must have the same set of L2 mappings and L2 attributes.

Enumeration: Availability of TDH.MEM.PAGE.PROMOTE version 1 is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDH.SYS.RD* (see 3.3.3.1). If

not supported, calling TDH.MEM.PAGE.PROMOTE with a version number higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

Support of NO_TRACK value of 1 is enumerated by TDX_FEATURES0.NON_BLOCKING_RESIZE (bit 35).

Blocking and TLB Tracking: If NO_TRACK is supported and its value is 1, the promoted GPA range is not checked for blocking and TLB tracking. The SEPT pages mapping that range are not removed; instead, they are converted to PT_TR pages which can later be either tracked and reclaimed or used as new SEPT pages for TDH.MEM.PAGE.DEMOTE for the current TD.

Else, if the TD may be running, the promoted GPA range must be blocked and TLB tracked. Else (e.g., TDH.MR.FINALIZE has not yet been executed, or the TD has been paused for export), no blocking and tracking is required.

Interruptibility: TDH.MEM.PAGE.PROMOTE is interruptible but not resumable. If a pending interrupt is detected during operation, TDH.MEM.PAGE.PROMOTE returns with a TDX_INTERRUPTED_RESTARTABLE status in RAX. No promote operation is done and no output operands except RAX are modified.

In such case, TDH.MEM.PAGE.PROMOTE should be invoked in a loop until it terminates successfully. The host VMM should be designed to avoid cases where interrupt storms prevent successful completion of TDH.MEM.PAGE.PROMOTE.

Removed Page Initialization: On platforms which do not use ACT, after the SEPT pages have been removed, the host VMM should initialize their content before they are reused as non-private pages, as described in the [Base Spec].

The table below shows the values of the SEPT HPA output arguments in RCX and R9 – R11 when version 1 or higher is selected and no error occurs (RAX returns TDX_SUCCESS).

Table 5.93: Meaning of TDH.MEM.PAGE.PROMOTE's SEPT HPA Arguments on Output (No Error)

Value	RCX on Output	R9 – R11 on Output
NULL_PA (-1)	N/A	No removed or converted SEPT page for this L2 VM
Valid HPA	L1 SEPT page HPA If NO_TRACK is supported and its value is 1, the page is converted to PT_TR. Else, the page is removed.	L2 SEPT page HPA If NO_TRACK is supported and its value is 1, the page is converted to PT_TR. Else, the page is removed.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.94: TDH.MEM.PAGE.PROMOTE Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[12]
Explicit	RCX	GPA and Level	Removed Secure EPT page	SEPT_PAGE	R	Private	$2^{12+9*Level}$ Bytes	Exclusive	None	None	Exclusive
Explicit	RDX	HPA	TDR page	TDR	RW	Opaque	4KB	Shared	Shared	Shared	None
Implicit	N/A	HPA	Merged HPA range	Blob	None	Private	N/A	Exclusive	None	None	None

[12] ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[12]
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A	None
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A	None
Implicit	N/A	GPA	L1 Secure EPT Tree	N/A	RW	Private	N/A	Shared	N/A	N/A	None
Implicit	N/A	GPA	Large page L1 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	N/A	N/A	None
Implicit	N/A	GPA	Small pages L1 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive	N/A	N/A	None
Implicit	N/A	GPA	L2 Secure EPT Trees	N/A	RW	Private	N/A	Shared(i)	N/A	N/A	None
Implicit	N/A	GPA	L2 Large page Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A	None
Implicit	N/A	GPA	L2 Small pages Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A	None

TDH.MEM.PAGE.PROMOTE checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must have been initialized or its metadata has been imported (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is either INITIALIZED, RUNNING, *_EXPORT, POST_IMPORT or LIVE_IMPORT).
5. The specified merged page level is either 1 (2MB) or 2 (1GB) – see 3.6.1 for a definition of EPT level.

If successful, the function does the following:

6. Walk the Secure EPT based on the GPA operand and locate the Secure EPT parent entry of the GPA range to be promoted to a merged large page.
7. Get the HPA of the Secure EPT page, which currently maps the GPA range to be promoted, from the Secure EPT above. Get its PAMT entry.
8. If NO_TRACK is 0, and the TD may run (its OP_STATE in either RUNNING, LIVE_EXPORT or LIVE_IMPORT):
   8.1. Check the Secure EPT entry: It must be a non-leaf, blocked (NL_BLOCKED) entry.
   8.2. Check that TLB tracking has been done, based on the above Secure EPT page's PAMT.BEPOCH.
9. Else (no blocking and tracking is required):
   9.1. Check the Secure EPT entry: It must be a non-leaf, mapped (NL_MAPPED) or blocked (NL_BLOCKED) entry.

If passed:

10. Scan the content of the above Secure EPT page and check all 512 entries:
    10.1. They are leaf entries (this also implies that the corresponding pages are PT_REG).
    10.2. Their state is either MAPPED or PENDING.
    10.3. They have contiguous HPA mapping aligned to the promoted range size.
    10.4. They have L2 mappings for the same set of L2 VMs.
11. For each L2 VM:

11.1. Walk the L2 Secure EPT based on the GPA operand and locate the L2 Secure EPT parent entry of the GPA range to be promoted to a merged large page.
11.2. If L2 mapping was found above to exist for this L2 VM:
11.2.1. The above SEPT walk should not fail. The L2 SEPT entry state is implicit: It must be a non-leaf, mapped (NL_MAPPED) or blocked (NL_BLOCKED) entry.
11.2.2. Scan the content of the above L2 Secure EPT page and check all 512 entries:
11.2.2.1. They should all have the same L2 attributes.
11.2.2.2. The following are implicit: All L2 SEPT entries are leaf entries, they all have the same state, and they have contiguous HPA mapping aligned to the promoted range size.
11.3. Else (L2 mapping was not found above to exist for this L2 VM):
11.3.1. The walk may fail; this is not considered an error.
11.3.2. If the walk succeeded, it implicitly arrives at an empty SEPT page.

If successful, the above checks imply that:

- The 2MB or 1GB GPA range to be promoted has a corresponding single HPA range and a single PAMT entry (PAMT_2M or PAMT_1G, respectively) owned by the current guest TD, and its current PAMT.PT is PAMT_NDA.
- The 512 child PAMT entries (PAMT_2M or PAMT_4K, respectively) of the above are owned by the current guest TD, and their PAMT.PT is PAMT_REG.

The function then does the following:

12. Merge the corresponding 512 physical pages into a single larger physical page:
12.1. Set the small page (PAMT_4K or PAMT_2M) entries state to PT_NDA.
12.2. Set the parent (PAMT_2M or PAMT_1G respectively) entry to PT_REG.
13. Atomically set the promoted Secure EPT entry to MAPPED or PENDING (depending on the small pages' Secure EPT entry state) leaf entry pointing to the merged HPA range.
14. If NO_TRACK is 0:
14.1. Remove the L1 Secure EPT page that previously mapped the 512 physical pages:
14.1.1. Atomically decrement TDR.CHLDCNT by 1.
14.1.1.1. Note that CHLDCNT counts the number of 4KB pages. The change is due only to the removal of the Secure EPT page.
14.1.2. On platforms which use ACT, overwrite the SEPT page content with the TD's random overwrite number, using MOVDIR64B, and clear the corresponding ACT bit.
14.1.3. Update the PAMT entry of the removed Secure EPT page to PT_NDA.
14.2. For each L2 VM where L2 mapping was found above to exist:
14.2.1. Remove the L2 Secure EPT page that previously mapped the 512 physical pages:
14.2.1.1. Atomically decrement TDR.CHLDCNT by 1.
14.2.1.2. On platforms which use ACT, overwrite the L2 SEPT page content with the TD's random overwrite number, using MOVDIR64B, and clear the corresponding ACT bit.
14.2.1.3. Update the PAMT entry of the removed Secure EPT page to PT_NDA.
15. Else (NO_TRACK is supported, and its value is 1), convert that Secure EPT pages that previously mapped the 512 physical pages to PT_TR pages:
15.1. Record the TD's TD_EPOCH in the L1 SEPT page's PAMT.BEPOCH.
15.2. Set the L1 SEPT page's PAMT.PT to PT_TR.
15.3. For each L2 VM where L2 mapping was found above to exist:
15.3.1. Record the TD's TD_EPOCH in the L2 SEPT page's PAMT.BEPOCH.
15.3.2. Set the L2 SEPT page's PAMT.PT to PT_TR.

Completion Status Codes

Table 5.95: TDH.MEM.PAGE.PROMOTE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_STATE_INCORRECT	
TDX_EPT_INVALID_PROMOTE_CONDITIONS	
TDX_EPT_WALK_FAILED	

Completion Status Code	Description
TDX_INTERRUPTED_RESTARTABLE	TDH.MEM.PAGE.PROMOTE's operation has been interrupted by an external event; it may be restarted (from its beginning) by calling it again.
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MEM.PAGE.PROMOTE is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	
TDX_TLB_TRACKING_NOT_DONE	

5.4.24. TDH.MEM.PAGE.RELOCATE Leaf

Relocate a 4KB mapped page from its current host physical address to another.

Table 5.96: TDH.MEM.PAGE.RELOCATE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the Secure EPT entry that maps the private page to be relocated, must be 0 (i.e., 4KB) (see 3.6.1).
	11:3	Reserved	Reserved: must be 0
	51:12	GPA	Bits 51:12 of the guest physical address of the private page to be relocated
	63:52	Reserved	Reserved: must be 0
RDX	Host physical address of the parent TDR page (HKID bits must be 0)		
R8	Host physical address of the relocated page target (HKID bits must be 0)		

Table 5.97: TDH.MEM.PAGE.RELOCATE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RCX	If TDH.MEM.PAGE.RELOCATE succeeded, RCX returns the HPA of the old physical page that has been removed (HKID bits are set to 0). In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page and may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. In other cases, RCX returns 0.
RDX	Extended error information part 2 In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.PAGE.RELOCATE replaces a mapped 4KB page mapping target HPA by moving the current page content to a new target HPA and updating the Secure-EPT mapping to the new target HPA. On successful operation, the previous mapped HPA target is marked is free in the PAMT.

Blocking and TLB Tracking: If the TD may be running, the relocated page must be blocked and TLB tracked. Else (e.g., TDH.MR.FINALIZE has not yet been executed, or the TD has been paused for export), no blocking and tracking is required.

Cache Lines Flushing (Future): On future platforms, if cache line flushing is required, as enumerated by TDX_FEATURES0.CLFLUSH_BEFORE_ALLOC (bit 23), then before calling TDH.MEM.PAGE.RELOCATE, the host VMM should ensure that no cache lines associated with the new physical page are in a Modified state, as described in the [Base Spec].

Removed Page Initialization: On platforms which do not use ACT, after the page has been relocated, the host VMM should initialize its content before it is reused as a non-private page, as described in the [Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.98: TDH.MEM.PAGE.RELOCATE Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[13]
Explicit	RCX	GPA and Level	TD private page	Blob	R	Private	4KB	Exclusive	None	None	Exclusive
Explicit	RDX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared	None
Explicit	R8	HPA	Target physical page	Blob	RW	Private	4KB	Exclusive	Shared	Shared	Exclusive
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A	None
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A	None
Implicit	N/A	GPA	L1 Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A	None
Implicit	N/A	GPA	L1 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	N/A	N/A	None
Implicit	N/A	GPA	L2 Secure EPT trees	N/A	RW	Private	N/A	Shared(i)	N/A	N/A	None
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A	None

TDH.MEM.PAGE.RELOCATE checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).

[13] ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

4. The TD must have been initialized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is either INITIALIZED or RUNNING).
5. The target page metadata in PAMT must be correct (PT must be PT_NDA).

If successful, the function does the following:

6. Walk the Secure EPT based on the GPA operand and level and find the currently mapped HPA.
7. If TLB tracking is required (based on the Secure EPT entry state and the TD's OP_STATE):
   7.1. Check that the SEPT entry is BLOCKED or PENDING_BLOCKED.
   7.2. Check that TLB tracking was done.

   Else:

   7.3. Check that the SEPT entry is MAPPED, BLOCKED, BLOCKEDW, EXPORTED*, PENDING, PENDING_BLOCKED, PENDING_BLOCKEDW or PENDING_EXPORTED*.
8. Check that the currently mapped HPA is different than the target HPA.

If successful, the function does the following:

9. On platforms which use ACT, set the target page's ACT bit to 1.
10. If the page state is not one of the PENDING* states, copy the currently mapped page content to the target page, using the TD's ephemeral private HKID and direct writes (MOVDIR64B).
11. On platforms which use ACT, overwrite the old page content with the TD's random overwrite number, using MOVDIR64B, and clear the corresponding ACT bit.
12. Free the currently mapped HPA by setting its PAMT.PT to PT_NDA.
13. Update the target page's PAMT entry with the PT_REG page type and the TDR physical address as the OWNER.
14. Update the Secure EPT entry:
    14.1. Set the HPA to point to the target page.
    14.2. Unblock the SEPT entry: if its state was BLOCKED or PENDING_BLOCKED, update it to MAPPED or PENDING, respectively.
15. For each L2 VM where the page has l2 mapping:
    15.1. Walk the L2 Secure EPT based on the GPA operand and find the leaf L2 SEPT entry mapping the page to be relocated.
    15.2. Update the L2 Secure EPT entry:
        15.2.1. Set the HPA to point to the target page.
        15.2.2. If the updated L1 SEPT entry state is one of the PENDING* states, set the L2 SEPT entry state to L2_BLOCKED. Else, set the L2 SEPT entry state to L2_MAPPED.

Completion Status Codes

Table 5.99: TDH.MEM.PAGE.RELOCATE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_WALK_FAILED	
TDX_GPA_RANGE_NOT_BLOCKED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MEM.PAGE.RELOCATE is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	

Completion Status Code	Description
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	
TDX_TLB_TRACKING_NOT_DONE	

5.4.25. TDH.MEM.PAGE.REMOVE Leaf

Remove a GPA-mapped 4KB, 2MB or 1GB private page from a TD.

Table 5.100: TDH.MEM.PAGE.REMOVE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version If the TDX module supports ACT, version may be 0 or 1. Else, version must be 0. See enumeration details below.
	63:24	Reserved	Must be 0
RCX	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the Secure EPT entry that maps the private page to be removed: either 0 (4KB), 1 (2MB) or 2 (1GB) – see 3.6.1.
	11:3	Reserved	Reserved: must be 0
	51:12	GPA	Bits 51:12 of the guest physical address of the private page to be removed
	63:52	Reserved	Reserved: must be 0
RDX	Host physical address of the parent TDR page (HKID bits must be 0)		

Table 5.101: TDH.MEM.PAGE.REMOVE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RCX	If TDH.MEM.PAGE.REMOVE succeeded, RCX returns the HPA of the removed page (HKID bits are set to 0). In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page and may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. In other cases, RCX returns 0.
RDX	Extended error information part 2 In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.PAGE.REMOVE removes a 4KB, 2MB or 1GB private page from the TD's Secure EPT tree (marks the SEPT entry as FREE). If the page is mapped in any L2 Secure EPT, the applicable L2 SEPT entries are marked as L2_FREE. On successful operation, TDH.MEM.PAGE.REMOVE marks the physical page as free in PAMT.

Enumeration: Availability of TDH.MEM.PAGE.REMOVE version 1 is enumerated by TDX_FEATURES0.ACT (bit 14), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.MEM.PAGE.REMOVE with version higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

Blocking and TLB Tracking: If the TD may be running, the removed page must be blocked and TLB tracked. Else (e.g., TDH.MR.FINALIZE has not yet been executed, or the TD has been paused for export), no blocking and tracking is required.

Removed Page Initialization: On platforms which do not use ACT, after the page has been removed, the host VMM should initialize its content before it is reused as non-private pages, as described in the [Base Spec].

Interruptibility: If called with version higher than 0, TDH.MEM.PAGE.REMOVE is interruptible and resumable. If a pending interrupt is detected during operation, TDH.MEM.PAGE.REMOVE returns with a TDX_INTERRUPTED_RESUMABLE status in RAX. No output operands except RAX are modified.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.102: TDH.MEM.PAGE.REMOVE Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[14]
Explicit	RCX	GPA and Level	TD private page	Blob	R	Private	$2^{12+9*Level}$ Bytes	Exclusive	None	None	Exclusive
Explicit	RDX	HPA	TDR page	TDR	RW	Opaque	4KB	Shared	Shared	Shared	None
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A	None
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A	None
Implicit	N/A	GPA	L1 Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A	None
Implicit	N/A	GPA	L1 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	N/A	N/A	None
Implicit	N/A	GPA	L2 Secure EPT trees	N/A	RW	Private	N/A	Shared(i)	N/A	N/A	None
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A	None

TDH.MEM.PAGE.REMOVE checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

[14] ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must be in one of the following states:
   4.1. The TD has been initialized locally by TDH.MNG.INIT.
   4.2. An import session is in progress, and either the out-of-order phase has started (TDH.IMPORT.TRACK has been successfully executed with an Epoch Token MBMD indicating a MIG_EPOCH value of 0xFFFFFFFF), or import has failed.
5. The specified level is either 0 (4KB), 1 (2MB) or 2 (1GB) – see 3.6.1 for a definition of EPT level.

If successful:

6. Walk the Secure EPT based on the GPA operand and find the leaf entry of the page to be removed.
7. If TLB tracking is required (based on the Secure EPT entry state and the TD's OP_STATE):
   7.1. Check that the SEPT entry is BLOCKED or PENDING_BLOCKED.
   7.2. Check that TLB tracking was done.

   Else:

   7.3. Check that the SEPT entry is MAPPED, BLOCKED, BLOCKEDW, PENDING, PENDING_BLOCKED or PENDING_BLOCKEDW.

If successful:

8. For each L2 VM where the page is mapped:
   8.1. Walk the L2 Secure EPT based on the GPA operand and find the page mapping to be removed.
   8.2. Set the L2 SEPT entry state to L2_FREE.
9. If an import session is in progress:
   9.1. Set the SEPT entry state to REMOVED.
   9.2. Record the migration epoch in the SEPT entry.
   Else:
   9.3. Set the SEPT entry state to FREE.
10. On platforms which do not use ACT:
    10.1. Atomically decrement TDR.CHLDCNT by 1, 512 or 512^2 depending on the removed TD private page size (4KB, 2MB or 1GB, respectively).
    10.2. Free the physical page: Set the PAMT entry of the removed TD private page to PT_NDA.
11. On platforms with ACT-protected memory:
    11.1. If the page size is 4KB:
        11.1.1. Atomically decrement TDR.CHLDCNT by 1.
        11.1.2. Overwrite the page using MOVDIR64B with the TD's random number overwrite value.
        11.1.3. Update the page's bit in ACT to shared.
        11.1.4. Free the physical page: Set the PAMT entry of the removed TD private page to PT_NDA.
    11.2. Else:
        11.2.1. On first iteration, identified by PAMT.PT other than PT_PR:
            11.2.1.1. Reset overwrite position to 0: Set PAMT.BEPOCH to 0.
            11.2.1.2. Mark the physical page as pending release: Set PAMT.PT to PT_PR.
        11.2.2. Overwrite the page content with the TD's random overwrite value using MOVDIR64B. Start from the offset value stored in PAMT.BEPOCH. Periodically, check for pending interrupts.
        11.2.3. If there is a pending interrupt, then:
            11.2.3.1. Save the last overwrite offset into PAMT.BEPOCH.
            11.2.3.2. Execute SFENCE.
            11.2.3.3. For version 0, execute SEAMRET without changing the VMM CPU state and without changing the RIP.
            11.2.3.4. For version 1, execute SEAMRET with TDX_INTERRUPTED_RESUMABLE status.
        If passed, page is filled with random data.
        11.2.4. Set the page's ACT bit(s) to 0.
        11.2.5. Atomically decrement TDR.CHLDCNT by 512 or 512^2 depending on the removed page size (2MB or 1GB, respectively).
        11.2.6. Update the PAMT entry of the reclaimed page to PT_NDA.

Completion Status Codes

Table 5.103: TDH.PHYMEM.PAGE.REMOVE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_WALK_FAILED	
TDX_GPA_RANGE_NOT_BLOCKED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MEM.PAGE.REMOVE is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	
TDX_TLB_TRACKING_NOT_DONE	

5.4.26. TDH.MEM.RANGE.BLOCK Leaf

Block a TD private GPA range (i.e., a Secure EPT page or a TD private page) at any level (4KB, 2MB, 1GB, 512GB, 256TB, etc.) from creating new GPA-to-HPA address translations.

Table 5.104: TDH.MEM.RANGE.BLOCK Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the Secure EPT entry that maps the GPA range to be blocked – see 3.6.1 Level must between 0 and 3 for a 4-level EPT or between 0 and 4 for a 5-level EPT.
	11:3	Reserved	Reserved: must be 0
	51:12	GPA	Bits 51:12 of the GPA range to be blocked Depending on the level, the following least significant bits must be 0: Level 0 (EPTE): None Level 1 (EPDE): Bits 20:12 Level 2 (EPDPTE): Bits 29:12 Level 3 (EPML4E): Bits 38:12 Level 4 (EPML5E): Bits 47:12
	63:52	Reserved	Reserved: must be 0
RDX	Host physical address of the parent TDR page (HKID bits must be 0)		

5

Table 5.105: TDH.MEM.RANGE.BLOCK Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page and may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. In other cases, RCX returns 0.

Operand	Description
RDX	Extended error information part 2 In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.RANGE.BLOCK finds the Secure EPT entry for the given GPA and level, and it marks it as blocked (BLOCKED or PENDING_BLOCKED as appropriate). It records the current TD's TLB epoch in the PAMT entry of the physical Secure EPT page or TD private page mapped by the blocked Secure EPT entry.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.106: TDH.MEM.RANGE.BLOCK Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA and Level	Secure EPT page or TD private page	Blob	None	Private	$2^{12+9*Level}$ Bytes	None	None	None
Explicit	RDX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT tree	N/A	RW	Private	N/A	Exclusive	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive(h)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT trees	N/A	RW	Private	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A

TDH.MEM.RANGE.BLOCK checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must have been initialized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is either INITIALIZED or RUNNING).
5. The specified level is of an EPT entry – i.e., 0 to 3 for 4-level EPT or 0 to 4 for 5-level EPT. See 3.6.1 for a definition of EPT level.

If successful, the function does the following:

6. Walk the Secure EPT based on the GPA operand and find the Secure EPT entry to be blocked.
7. Check the Secure EPT entry is not free and not blocked (its state should be NL_MAPPED, MAPPED or PENDING).

If passed:

8. Block the Secure EPT entry. Set its state to NL_BLOCKED (if it was NL_MAPPED), BLOCKED (if it was MAPPED) or PENDING_BLOCKED (if it was PENDING).
9. For each L2 VM where there is an SEPT entry for the given GPA and level:
   9.1. Walk the L2 Secure EPT based on the GPA and level operand and find the L2 SEPT entry to be blocked.
   9.2. If the L1 SEPT entry is a leaf entry, then set the L2 SEPT entry state to L2_BLOCKED and save its attributes. Note that if the page was blocked for writing, then the W bit has already been saved.
   9.3. Else (L1 SEPT entry is a non-leaf, mapping an SEPT page), set the non-leaf L2 SEPT entry state to NL_BLOCKED.
   Note: There is no need to write TD_EPOCH to the L2 SEPT page's PAMT. Blocked epoch is implicit from the L1 SEPT page.

If passed:

10. Read the TD's epoch (TDCS.TD_EPOCH) and write it to the PAMT entry of the blocked Secure EPT page or TD private page (PAMT.BEPOCH).

Completion Status Codes

Table 5.107: TDH.MEM.RANGE.BLOCK Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_STATE_INCORRECT	
TDX_EPT_WALK_FAILED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MEM.RANGE.BLOCK is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.27. TDH.MEM.RANGE.UNBLOCK Leaf

Remove the blocking of a TD private GPA range (i.e., a Secure EPT page or a TD private page), at any level (4KB, 2MB, 1GB, 512GB, 256TB etc.) previously blocked by TDH.MEM.RANGE.BLOCK.

Table 5.108: TDH.MEM.RANGE.UNBLOCK Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the Secure EPT entry that maps the GPA range to be unblocked – see 3.6.1 Level must between 0 and 3 for a 4-level EPT or between 0 and 4 for a 5-level EPT.
	11:3	Reserved	Reserved: must be 0
	51:12	GPA	Bits 51:12 of the guest physical address range to be unblocked Depending on the level, the following least significant bits must be 0: Level 0 (EPTE): None Level 1 (EPDE): Bits 20:12 Level 2 (EPDPTE): Bits 29:12 Level 3 (EPML4E): Bits 38:12 Level 4 (EPML5E): Bits 47:12
	63:52	Reserved	Reserved: must be 0
RDX	Host physical address of the parent TDR page (HKID bits must be 0)		

Table 5.109: TDH.MEM.RANGE.UNBLOCK Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page and may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. In other cases, RCX returns 0.

Operand	Description
RDX	Extended error information part 2 In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.RANGE.UNBLOCK finds the blocked Secure EPT entry for the given GPA and level. It checks that the entry has been blocked and TLB tracking has been done, and then it marks the entry as non-blocked (MAPPED or PENDING as appropriate).

Blocking and TLB Tracking: If the TD may be running, the unblocked GPA range must be blocked and TLB tracked. Else (e.g., TDH.MR.FINALIZE has not yet been executed, or the TD has been paused for export), no blocking and tracking is required.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.110: TDH.MEM.RANGE.UNBLOCK Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA and Level	Secure EPT page or TD private page	Blob	None	Private	$2^{12+9*Level}$ Bytes	None	None	None
Explicit	RDX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT tree	N/A	RW	Private	N/A	Exclusive	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive(h)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT tree	N/A	RW	Private	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A

TDH.MEM.RANGE.UNBLOCK checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).

4. The TD must have been initialized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is either INITIALIZED or RUNNING).
5. The specified level is of an EPT entry (i.e., 0 to 3 for 4-level EPT or 0 to 4 for 5-level EPT) – see 3.6.1 for a definition of EPT level.

If successful, the function does the following:

6. Walk the Secure EPT based on the GPA operand and find the Secure EPT page or TD private page to be unblocked.
7. Check the page's parent Secure EPT entry is blocked (NL_BLOCKED, BLOCKED or PENDING_BLOCKED).
8. If TLB tracking is required (based on the Secure EPT entry state and the TD's OP_STATE):
   8.1. Check that TLB tracking was done.

If successful, the function does the following:

9. For each L2 VM where there is an SEPT entry for the given GPA and level:
   9.1. Walk the L2 Secure EPT based on the GPA operand and find the L2 SEPT entry to be unblocked.
   9.2. If the updated L1 SEPT entry is a leaf entry, and its state is not one of the PENDING* states, set the L2 SEPT entry state to L2_MAPPED and restore the L2 SEPT attributes.
   9.3. Else (L1 SEPT entry is a non-leaf, mapping an SEPT page), set the non-leaf L2 SEPT entry state to NL_MAPPED.
10. Unblock the Secure EPT entry. Atomically set its state to NL_MAPPED (if it was NL_BLOCKED), MAPPED (if it was BLOCKED) or PENDING (if it was PENDING_BLOCKED).

Completion Status Codes

Table 5.111: TDH.MEM.RANGE.UNBLOCK Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_WALK_FAILED	
TDX_GPA_RANGE_NOT_BLOCKED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MEM.RANGE.UNBLOCK is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	
TDX_TLB_TRACKING_NOT_DONE	

5.4.28. TDH.MEM.RD Leaf

Read a 64b chunk from a debuggable guest TD private memory.

Table 5.112: TDH.MEM.RD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The guest physical address of a naturally aligned 8-byte chunk of a guest TD private page		
RDX	Host physical address of the parent TDR page (HKID bits must be 0)		

Table 5.113: TDH.MEM.RD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page and may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. In other cases, RCX returns 0.
RDX	Extended error information part 2 In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0.
R8	Content of the memory chunk In case of an error, as indicated by RAX, R8 returns 0
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.RD reads a 64b chunk from a debuggable guest TD private memory.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.114: TDH.MEM.RD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA	TD private memory	Blob	R	Private	8B	None	None	None
Explicit	RDX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	R	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT tree	N/A	R	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	R	Private	N/A	Exclusive	N/A	N/A

TDH.MEM.RD checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.KEY_STATE is TD_KEYS_CONFIGURED).
4. TDCS must have been initialized (TDR.INIT is TRUE).
5. The TD is debuggable (TDCS.ATTRIBUTES.DEBUG is 1).

If successful, the function does the following:

6. Walk the Secure EPT based on the GPA operand and find the leaf entry.
7. Check that the Secure EPT entry state is PRESENT.

If passed:

8. Read the content of the memory chunk.

Completion Status Codes

Table 5.115: TDH.MEM.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_NOT_PRESENT	
TDX_EPT_ENTRY_STATE_INCORRECT	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NON_DEBUG	

Completion Status Code	Description
TDX_TDCS_NOT_ALLOCATED	

Intel TDX Application Binary Interface (ABI) Reference

5.4.29. TDH.MEM.SEPT.ADD Leaf

Add and map 4KB L1 and L2 Secure EPT pages to a TD.

Table 5.116: TDH.MEM.SEPT.ADD Input Operands Definition

<table>
<tr><th>Operand</th><th colspan="3">Description</th></tr>
<tr><td rowspan="5">RAX</td><td colspan="3">SEAMCALL instruction leaf number and version, see 5.4.1</td></tr>
<tr><td>Bits</td><td>Field</td><td>Description</td></tr>
<tr><td>15:0</td><td>Leaf Number</td><td>Selects the SEAMCALL interface function</td></tr>
<tr><td>23:16</td><td>Version Number</td><td>Selects the SEAMCALL interface function version</td></tr>
<tr><td>63:24</td><td>Reserved</td><td>Must be 0</td></tr>
<tr><td rowspan="6">RCX</td><td colspan="3">EPT mapping information:</td></tr>
<tr><td>Bits</td><td>Name</td><td>Description</td></tr>
<tr><td>2:0</td><td>Level</td><td>Level of the non-leaf Secure EPT entry that will map the new Secure EPT page – see 3.6.1
Level must between 1 and 3 for a 4-level EPT or between 1 and 4 for a 5-level EPT.</td></tr>
<tr><td>11:3</td><td>Reserved</td><td>Reserved: must be 0</td></tr>
<tr><td>51:12</td><td>GPA</td><td>Bits 51:12 of the guest physical address of to be mapped for the new Secure EPT page
Depending on the level, the following least significant bits must be 0:
Level 1 (EPT): Bits 20:12
Level 2 (EPD): Bits 29:12
Level 3 (EPDPT): Bits 38:12
Level 4 (EPML4): Bits 47:12</td></tr>
<tr><td>63:52</td><td>Reserved</td><td>Reserved: must be 0</td></tr>
<tr><td rowspan="5">RDX</td><td colspan="3">TD handle and flags:</td></tr>
<tr><td>Bits</td><td>Name</td><td>Description</td></tr>
<tr><td>0</td><td>ALLOW_EXISTING</td><td>Flags that TDH.MEM.SEPT.ADD should not fail if an SEPT page to be added already exists in the L1 or L2 SEPT tree. Instead, it should just return an indication in the output operand, as described below.</td></tr>
<tr><td>11:1</td><td>Reserved</td><td>Reserved: must be 0</td></tr>
<tr><td>51:12</td><td>TDR_HPA</td><td>Bits 51:12 of the host physical address of the parent TDR page (HKID bits must be 0)</td></tr>
<tr><td>63:52</td><td>Reserved</td><td>Reserved: must be 0</td></tr>
<tr><td>R8</td><td colspan="3">Host physical address of the new L1 Secure EPT page to be added to the TD (HKID bits must be 0)
For TDH.MEM.SEPT.ADD version 1 or higher:
• If the value of R8 is NULL_PA (-1), no L1 SEPT page is added.
• Else, bit 63 of R8 is ignored.</td></tr>
</table>

Operand	Description
R9	For TDH.MEM.SEPT.ADD version 1 or higher, R9 specifies the HPA of a new L2 VM #1 Secure EPT page to be added to the TD (HKID bits must be 0). • If the value of R9 is NULL_PA (-1), no L2 VM #1 SEPT page is added. • Else, bit 63 of R9 is ignored.
R10	For TDH.MEM.SEPT.ADD version 1 or higher, R10 specifies the HPA of a new L2 VM #2 Secure EPT page to be added to the TD (HKID bits must be 0). • If the value of R10 is NULL_PA (-1), no L2 VM #2 SEPT page is added. • Else, bit 63 of R10 is ignored.
R11	For TDH.MEM.SEPT.ADD version 1 or higher, R11 specifies the HPA of a new L2 VM #3 Secure EPT page to be added to the TD (HKID bits must be 0). • If the value of R11 is NULL_PA (-1), no L2 VM #3 SEPT page is added. • Else, bit 63 of R11 is ignored.

Table 5.117: TDH.MEM.SEPT.ADD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RCX	In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RCX is unmodified. Else, RCX returns extended error information part 1. In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page; it may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. In other cases, RCX returns 0.
RDX	In case of an interruption, as indicated by RAX returning TDX_INTERRUPTED_RESUMABLE, RDX is unmodified. Else, RDX returns extended error information part 2. In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0.
R8	For TDH.MEM.SEPT.ADD version 1 or higher: • If a provided L1 SEPT page has been added, R8 returns NULL_PA (-1). • Else, if an L1 SEPT page already exists, bit 63 of R8 is set to 1, other bits are unmodified. • Else, bit 63 of R8 is cleared to 0, other bits are unmodified. For TDH.MEM.SEPT.ADD version 0, R8 is unmodified.
R9	For TDH.MEM.SEPT.ADD version 1 or higher: • If a provided L2 VM #1 SEPT page has been added, R9 returns NULL_PA (-1). • Else, if an L2 VM #1 SEPT page already exists, bit 63 of R9 is set to 1, other bits are unmodified. • Else, bit 63 of R9 is cleared to 0, other bits are unmodified. For TDH.MEM.SEPT.ADD version 0, R9 is unmodified.

Operand	Description
R10	For TDH.MEM.SEPT.ADD version 1 or higher: • If a provided L2 VM #2 SEPT page has been added, R10 returns NULL_PA (-1). • Else, if an L2 VM #2 SEPT page already exists, bit 63 of R10 is set to 1, other bits are unmodified. • Else, bit 63 of R10 is cleared to 0, other bits are unmodified. For TDH.MEM.SEPT.ADD version 0, R10 is unmodified.
R11	For TDH.MEM.SEPT.ADD version 1 or higher: • If a provided L2 VM #3 SEPT page has been added, R11 returns NULL_PA (-1). • Else, if an L2 VM #3 SEPT page already exists, bit 63 of R11 is set to 1, other bits are unmodified. • Else, bit 63 of R11 is cleared to 0, other bits are unmodified. For TDH.MEM.SEPT.ADD version 0, R11 is unmodified.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.SEPT.ADD adds a set of 4KB Secure EPT pages to a TD and maps them to the provided GPA and level. SEPT pages can be added to the main (L1) SEPT tree and/or to one or more of the L2 VMs' SEPT trees. TDH.MEM.SEPT.ADD initializes the SEPT pages to hold 512 free entries using the TD's ephemeral private key.

L2 SEPT trees may not be deeper than the L1 SEPT tree. To add an L2 SEPT page at some level, there must either already be an L1 SEPT page at that level, or an L1 SEPT page at that level is being added by the current TDH.MEM.SEPT.ADD invocation.

Enumeration: Availability of TDH.MEM.SEPT.ADD version 1 is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.MEM.SEPT.ADD with a version number higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

Interruptibility: If a version number higher than 0 is specified on input, TDH.MEM.SEPT.ADD is interruptible. If a pending interrupt is detected during operation, TDH.MEM.SEPT.ADD returns with a TDX_INTERRUPTED_RESUMABLE status in RAX. The SEPT page HPA values in R8, R9, R10 and R11 are updated.

TDH.MEM.SEPT.ADD is designed to be invoked in a loop until all required SEPT pages have been added:

1. Call TDH.MEM.SEPT.ADD.
2. While RAX indicates TDX_INTERRUPTED_RESUMABLE:
   2.1. Call TDH.MEM.SEPT.ADD with the GPR values as returned by the previous call.
   2.2. If an error indication other than TDX_INTERRUPTED_RESUMABLE is returned, abort.

The table below shows the values of the SEPT HPA arguments in R8 – R11 when version 1 or higher is selected and no error occurs (RAX returns TDX_SUCCESS or TDX_INTERRUPTED_RESUMABLE).

Table 5.118: Meaning of TDH.MEM.SEPT.ADD's SEPT HPA Arguments on Input and Output (Version > 0, No Error)

Value	R8 – R11 on Input	R8 – R11 on Output
NULL_PA (-1)	No new SEPT page to add	New SEPT page has been added
Bits 62:0: Valid HPA, HKID bits are 0 Bit 63: 0	Bits 62:0: HPA of new SEPT page to add Bit 63: Ignored	N/A
Bits 62:0: Valid HPA, HKID bits are 0 Bit 63: 1		SEPT page already exists, new SEPT page has not been used

Atomicity: Unless terminated with a TDX_INTERRUPTED_RESUMABLE indication, TDH.MEM.SEPT.ADD either fully succeeds in adding the requested SEPT pages or doesn't add any page.

In case of an interrupt (TDX_INTERRRUPTED_RESUMABLE), if the host VMM invokes TDH.MEM.SEPT.ADD in a loop as described above and doesn't initiate other operations that impact TDH.MEM.SEPT.ADD (e.g., TDH.MEM.RANGE.BLOCK), then this atomicity still holds at the end of the loop.

Cache Lines Flushing (Future): On future platforms, if cache line flushing is required, as enumerated by TDX_FEATURES0.CLFLUSH_BEFORE_ALLOC (bit 23), then before calling TDH.MEM.SEPT.ADD, the host VMM should ensure that no cache lines associated with the added SEPT physical pages are in a Modified state, as described in the [Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.119: TDH.MEM.SEPT.ADD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA and Level	Secure EPT page (GPA)[15]	SEPT_PAGE	RW	Private	$2^{12+9*Level}$ Bytes	N/A	N/A	N/A
Explicit	RDX	HPA	TDR page	TDR	RW	Opaque	4KB	Shared	Shared	Shared
Explicit	R8	HPA	Secure EPT page (HPA)[15]	SEPT_PAGE	RW	Private	4KB	Exclusive	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT trees	N/A	RW	Private	N/A	Shared(i)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A

TDH.MEM.SEPT.ADD checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must have been initialized by TDH.MNG.INIT and not paused by THH.EXPORT.PAUSE, or an import session is in progress, started by TDH.IMPORT.STATE.IMMUTABLE and not failed.

[15] RCX and R8 denote the same Secure EPT page operand, using HPA and GPA respectively

5. The specified level is of an EPT non-leaf entry – i.e., 1 to 3 for 4-level EPT or 1 to 4 for 5-level EPT. See 3.6.1 for a definition of EPT level.

If successful, the function does the following:

6. SEPT physical pages checks:
   - 6.1. Check that no non-NULL_PA address has been provided in R9, R10 or R11 for a non-existing L2 VM.
   - 6.2. If the version number indicated in RAX is 1 or higher, check and lock the new SEPT physical pages:
     - 6.2.1. For each of the SEPT page HPAs provided in R8, R9, R10 and R11, if the value is not NULL_PA, lock the PAMT entry and check that the page metadata is correct (PT must be PT_NDA).
   - 6.3. Else (version is 0):
     - 6.3.1. For the L1 SEPT page HPA provided in R8, lock the PAMT entry and check that the page metadata is correct (PT must be PT_NDA).

If passed:

7. SEPT trees walk and state checks:
   - 7.1. Walk the L1 Secure EPT based on the requested GPA and level and find the SEPT entry.

   If L1 SEPT walk succeeded:

   - 7.2. If requested to add an L1 SEPT page (i.e., R8 is a not a NULL_PA):
     - 7.2.1. The L1 SEPT entry state should either be FREE,
     - 7.2.2. Or, if ALLOW_EXISTING is 1, the L1 SEPT entry state can be NL_MAPPED (i.e., there's already an L1 SEPT page at the requested level). In this case, mark the return value in R8 to indicate that the new L1 SEPT page was not used.
     - 7.2.3. On other L1 SEPT entry states, abort with an error indication in RAX.
   - 7.3. Else (no L1 SEPT page is to be added):
     - 7.3.1. Check that an L1 SEPT page exists (the L1 SEPT entry state is NL_MAPPED).

   If passed:

   - 7.4. If the version number indicated in RAX is 1 or higher, then for each L2 VM do the following:
     - 7.4.1. Walk the L2 Secure EPT based on the requested GPA and level and find the L2 SEPT entry.

     If L2 SEPT walk succeeded:

     - 7.4.2. The L2 SEPT entry state should either be L2_FREE,
     - 7.4.3. Or, if ALLOW_EXISTING is 1, the L2 SEPT entry state can be L2_NL_MAPPED (i.e., there's already an L2 SEPT page at the requested level). In this case, mark the return value in R9, R10 or R11 to indicate that the new SEPT page was not used.
     - 7.4.4. On other L2 SEPT entry states, abort with an error indication in RAX.

If passed:

8. SEPT page additions:
   - 8.1. If the L1 SEPT entry state is FREE:
     - 8.1.1. On platforms which use ACT, set the new L1 SEPT page's bit in ACT to 1.
     - 8.1.2. Initialize the new L1 Secure EPT page, indicating 512 entries in the FREE state, using the TD's ephemeral private HKID and direct writes (MOVDIR64B).
     - 8.1.3. Update the parent L1 Secure EPT entry with the new Secure EPT page HPA and NL_MAPPED state.
     - 8.1.4. Increment TDR.CHLDCNT.
     - 8.1.5. Update the new Secure EPT page's PAMT entry with the PT_EPT page type and the TDR physical address as the OWNER.
     - 8.1.6. If the version number indicated in RAX is 1 or higher, set the returned value of R8 to NULL_PA, indicating that a new SEPT page has been added.

   If passed:

   - 8.2. If the version number indicated in RAX is 1 or higher, then for each L2 VM do the following:
     - 8.2.1. If the L2 SEPT entry state is L2_FREE:
       - 8.2.1.1. If an interrupt is pending, abort with a TDX_INTERRUPTED_RESUMABLE status.
       - 8.2.1.2. On platforms which use ACT, set the new L2 SEPT page's bit in ACT to 1.
       - 8.2.1.3. Initialize the new L2 Secure EPT page, indicating 512 entries in the L2_FREE state, using the TD's ephemeral private HKID and direct writes (MOVDIR64B).
       - 8.2.1.4. Update the parent Secure EPT entry with the new Secure EPT page HPA and NL_MAPPED state.
       - 8.2.1.5. Increment TDR.CHLDCNT.

8.2.1.6. Update the new L2 Secure EPT page's PAMT entry with the PT_EPT page type and the TDR physical address as the OWNER.
8.2.1.7. Set the returned value of R9, R10 or R11 to NULL_PA, indicating that a new L2 SEPT page has been added.

Completion Status Codes

Table 5.120: TDH.MEM.SEPT.ADD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_STATE_INCORRECT	
TDX_EPT_WALK_FAILED	
TDX_INTERRUPTED_RESUMABLE	TDH.MEM.SEPT.ADD's operation has been interrupted by an external event; it may be resumed from the point it was interrupted by calling it again.
TDX_L2_SEPT_ENTRY_NOT_FREE	
TDX_L2_SEPT_WALK_FAILED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MEM.SEPT.ADD is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.30. TDH.MEM.SEPT.RD Leaf

Read a Secure EPT entry.

Table 5.121: TDH.MEM.SEPT.RD Input Operands Definition

Operand		Description		
RAX	Leaf and Version	SEAMCALL instruction leaf number and version, see 5.4.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the SEAMCALL interface function
		23:16	Version Number	Selects the SEAMCALL interface function version
		63:24	Reserved	Must be 0
RCX	GPA Mapping	EPT mapping information:		
		Bits	**Name**	**Description**
		2:0	Level	Level of the Secure EPT entry to read – see 3.6.1 Level must between 0 and 3 for a 4-level EPT or between 0 and 4 for a 5-level EPT.
		11:3	Reserved	Reserved: must be 0
		51:12	GPA	Bits 51:12 of the guest physical address for the Secure EPT entry to read Depending on the level, the following least significant bits must be 0: Level 0 (EPTE): None Level 1 (EPDE): Bits 20:12 Level 2 (EPDPTE): Bits 29:12 Level 3 (EPML4E): Bits 38:12 Level 4 (EPML5E): Bits 47:12
		63:52	Reserved	Reserved: must be 0
RDX	TD Handle and Flags	TD handle and flags:		
		Bits	**Name**	**Description**
		0	READ_L2_ATTR	Flags that L2 attributes should be returned in R8
		11:1	Reserved	Reserved: must be 0
		51:12	HPA	Bits 51:12 of the host physical address of the parent TDR page (HKID bits must be 0)
		63:52	Reserved	Reserved: must be 0

Table 5.122: TDH.MEM.SEPT.RD Output Operands Definition

Operand		Description
RAX	Status	SEAMCALL instruction return code – see 5.4.1

Operand		Description
RCX	SEPT Entry	Secure EPT entry architectural content – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page; it may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. • In case of successful operation, the requested entry's architectural content is returned. • In case of EPT walk error, the architectural content of the Secure EPT entry where the error was detected is returned. In other cases, RCX returns 0.
RDX	SEPT Level and State	Secure EPT entry level and state – see 3.6.2 • In case of successful operation, the requested entry's information is returned. • In case of EPT walk error, the information of the Secure EPT entry where the error was detected is returned. In other cases, RDX returns 0.
R8	L2 Attributes	If the TD's ATTRIBUTES.DEBUG is 1 and READ_L2_ATTR is 1, R8 returns the L2 attributes of the applicable L2 SEPT entries, in the format defined in 3.6.3. Else, R8 is unmodified.
Other	N/A	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.SEPT.RD reads a Secure EPT entry. If the TD's ATTRIBUTES.DEBUG is 1, then TDH.MEM.SEPT.RD can return the page's L2 attributes.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.123: TDH.MEM.SEPT.RD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA and Level	Secure EPT entry	SEPT_ENTRY	R	Private	$2^{12+9*Level}$ Bytes	None	None	None
Explicit	RDX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT Tree	N/A	R	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	N/A	N/A

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	GPA	L2 Secure EPT Tree	N/A	R	Private	N/A	Shared(i)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A

TDH.MEM.SEPT.RD checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must have been initialized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is either INITIALIZED or RUNNING).
5. The specified level is of an EPT entry (i.e., 0 to 3 for 4-level EPT or 0 to 4 for 5-level EPT) – see 3.6.1 for a definition of EPT level.

If successful, the function does the following:

6. If READ_L2_ATTR is set, check that TDCS.ATTRIBUTES.DEBUG is 1.

If passed:

7. Walk the L1 Secure EPT based on the GPA and level operand and find the Secure EPT entry.
8. Read the L1 Secure EPT entry contents.
9. If READ_L2_ATTR is set:
   - 9.1. If the L1 SEPT entry is a leaf entry, then for each L2 VM where the page is mapped
     - 9.1.1. Walk the L2 SEPT based on the GPA and level operand and find the L2 SEPT entry.
     - 9.1.2. Read the leaf L2 SEPT entry and build the L2 attributes to be returned.
   - 9.2. If the L1 SEPT entry is a non-leaf entry, then for each L2 VM:
     - 9.2.1. Walk the L2 SEPT based on the GPA and level operand, and find the non-leaf L2 SEPT entry, if any.
     - 9.2.2. Read the non-leaf L2 SEPT entry and build the L2 attributes to be returned.
   - 9.3. Else (the L1 SEPT entry is FREE):
     - 9.3.1. Return 0 as the L2 attributes.

Completion Status Codes

Table 5.124: TDH.MEM.SEPT.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_WALK_FAILED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MEM.SEPT.RD is successful.
TDX_SYS_NOT_READY	

Completion Status Code	Description
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.31. TDH.MEM.SEPT.REMOVE Leaf

Remove an empty L1 Secure EPT page and any associated L2 SEPT pages from a TD.

Table 5.125: TDH.MEM.SEPT.REMOVE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Versions 0 and 1 are supported. See the enumeration details below.
	63:24	Reserved	Must be 0
RCX	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the non-leaf Secure EPT entry that maps the Secure EPT page to be removed – see 3.6.1 Level must be between 1 and 3 for a 4-level EPT or between 1 and 4 for a 5-level EPT.
	11:3	Reserved	Reserved: must be 0
	51:12	GPA	Bits 51:12 of the guest physical address for the Secure EPT page to be removed Depending on the level, the following least significant bits must be 0: Level 1 (EPT): Bits 20:12 Level 2 (EPD): Bits 29:12 Level 3 (EPDPT): Bits 38:12 Level 4 (EPML4): Bits 47:12
	63:52	Reserved	Reserved: must be 0
RDX	Host physical address of the parent TDR page (HKID bits must be 0)		

Table 5.126: TDH.MEM.SEPT.REMOVE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RCX	If TDH.MEM.SEPT.REMOVE succeeded, RCX returns the HPA of the removed SEPT page (HKID bits are set to 0). In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page and may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. In other cases, RCX returns 0.

Operand	Description
RDX	Extended error information part 2 In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0.
R9	If TDH.MEM.SEPT.REMOVE version is 1 or higher: • If L2 VM #1's L2 SEPT page has been removed, R9 returns the HPA of that SEPT page (HKID bits are set to 0). • Else, R9 returns NULL_PA (-1). Else, R9 is unmodified.
R10	If TDH.MEM.SEPT.REMOVE version is 1 or higher: • If L2 VM #2's L2 SEPT page has been removed, R10 returns the HPA of that SEPT page (HKID bits are set to 0). • Else, R10 returns NULL_PA (-1). Else, R10 is unmodified.
R11	If TDH.MEM.SEPT.REMOVE version is 1 or higher: • If L2 VM #3's L2 SEPT page has been removed, R11 returns the HPA of that SEPT page (HKID bits are set to 0). • Else, R11 returns NULL_PA (-1). Else, R11 is unmodified.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.SEPT.REMOVE removes an empty Secure EPT page or pages, with all 512 entries marked as FREE, from the TD's Secure EPT trees.

The L1 SEPT page and all existing L2 SEPT pages at the requested GPA and level are removed.

On successful operation, it TDH.MEM.SEPT.REMOVE marks the removed 4KB physical pages as free in PAMT.

Enumeration: Availability of TDH.MEM.SEPT.REMOVE version 1 is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.MEM.SEPT.REMOVE with a version number higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

Blocking and TLB Tracking: If the TD may be running, the removed GPA range must be blocked and TLB tracked. Else (e.g., TDH.MR.FINALIZE has not yet been executed, or the TD has been paused for export), no blocking and tracking is required.

Atomicity: TDH.MEM.SEPT.REMOVE either fully succeeds in removing the requested SEPT pages or doesn't remove any page.

Removed Page Initialization: On platforms which do not use ACT, after the SEPT pages have been removed, the host VMM should initialize their content before they are reused as non-private pages, as described in the [Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.127: TDH.MEM.SEPT.REMOVE Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA and Level	Secure EPT page	SEPT_PAGE	R	Private	$2^{12+9*Level}$ Bytes	Exclusive	None	None
Explicit	RDX	HPA	TDR page	TDR	RW	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT Tree	N/A	RW	Private	N/A	Exclusive	N/A	N/A
Implicit	N/A	GPA	L1 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive(h)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT Trees	N/A	RW	Private	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A

TDH.MEM.SEPT.REMOVE checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must either have been initialized by TDH.MNG.INIT, or an import session has begun by TDH.IMPORT.STATE.IMMUTABLE.
5. The specified level is of a non-leaf EPT entry (i.e., 1 to 3 for 4-level EPT or 1 to 4 for 5-level EPT) – see 3.6.1 for a definition of EPT level.

If successful, the function does the following:

6. Walk the L1 Secure EPT based on the GPA operand and find the non-leaf SEPT entry of the SEPT page to be removed.
7. If TLB tracking is required (based on the Secure EPT entry state and the TD's OP_STATE):
   7.1. Check the L1 Secure EPT entry is a non-leaf blocked (NL_BLOCKED) entry.
   7.2. Check that TLB tracking was done.
8. Scan the L1 Secure EPT page content and check all 512 entries are FREE.

If passed:

9. For each L2 VM:
   9.1. Walk the L2 Secure EPT based on the GPA and level operand and find the applicable non-leaf SEPT entry.
   9.2. If an L2 SEPT entry was found, and its state is not FREE:
      9.2.1. Atomically decrement TDR.CHLDCNT.
      9.2.2. On platforms which use ACT, overwrite the removed L2 SEPT page with the TD's random overwrite number using MOVDIR64B.
      9.2.3. Set the PAMT entry of the removed L2 SEPT page to PT_NDA.
      9.2.4. Set the parent L2 Secure EPT entry to FREE.

If passed:

10. Set the parent L1 Secure EPT entry to FREE.
11. Atomically decrement TDR.CHLDCNT.

12. On platforms which use ACT, overwrite the removed L1 SEPT page with the TD's random overwrite number using MOVDIR64B.
13. Set the PAMT entry of the removed L1 Secure EPT page to PT_NDA.

Completion Status Codes

Table 5.128: TDH.MEM.SEPT.REMOVE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_STATE_INCORRECT	
TDX_EPT_WALK_FAILED	
TDX_GPA_RANGE_NOT_BLOCKED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MEM.SEPT.REMOVE is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	
TDX_TLB_TRACKING_NOT_DONE	

5.4.32. TDH.MEM.SHARED.SEPT.WR Leaf

Add mapping of a Shared GPA range from Secure EPT into Shared EPT pages.

Table 5.129: TDH.MEM.SHARED.SEPT.WR Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Versions 0 and 1 are supported. See the enumeration details below.
	63:24	Reserved	Must be 0
RCX	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the Secure EPT entry to write Level must be the level of the Secure EPT page which is indexed the GPA Shared bit (i.e., GPAW + 3).
	11:3	Reserved	Reserved: must be 0
	51:12	GPA	Bits 51:12 of the shared guest physical address for the Secure EPT entry to write. Depending on the level, the following least significant bits must be 0: Level 3 (EPML4E): Bits 38:12 Level 4 (EPML5E): Bits 47:12
	63:52	Reserved	Reserved: must be 0
RDX	Host physical address of the parent TDR page (HKID bits must be 0)		
R8	If the requested version is 1 or higher and the value of R8 is NULL_PA (-1), R8 is ignored. Else, R8 specifies a value to be written to the applicable L1 EPT entry.		
R9	If the requested version is 0, R9 is ignored. Else (the requested version is 1 or higher), the number of L2 VMs is >= 1 and the value of R9 is not NULL_PA (-1), R9 specifies a value to be written to the applicable L2 VM #1 Secure EPT entry. Else, R9 is ignored.		
R10	If the requested version is 0, R10 is ignored. Else (the requested version is 1 or higher), the number of L2 VMs is >= 2 and the value of R10 is not NULL_PA (-1), R10 specifies a value to be written to the applicable L2 VM #2 Secure EPT entry. Else, R10 is ignored.		
R11	If the requested version is 0, R11 is ignored. Else (the requested version is 1 or higher), the number of L2 VMs is >= 3 and the value of R11 is not NULL_PA (-1), R11 specifies a value to be written to the applicable L2 VM #3 Secure EPT entry. Else, R11 is ignored.		

Table 5.130: TDH.MEM.SHARED.SEPT.WR Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page; it may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. In other cases, RCX returns 0.
RDX	Extended error information part 2 In case of EPT walk error, Secure EPT entry level and state where the error was detected – see 3.6.2 In other cases, RDX returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

Enumeration: Availability of TDH.MEM.SHARED.SEPT.WR is enumerated by TDX_FEATURES0.TDX_CONNECT (bit 6), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.MEM.SHARED.SEPT.WR returns a TDX_OPERAND_INVALID(RAX) status.

Availability of TDH.MEM.SHARED.SEPT.WR version 1 or higher is enumerated by TDX_FEATURES0.TDX_CONNECT_PARTITIONING (bit 32).

TDH.MEM.SHARED.SEPT.WR enables the host VMM to write Secure EPT entries associated with Shared EPT, to map Shared EPT pages. This is required to allow TDX Connect devices DMA translations of shared GPAs using the Secure EPT root page. Only SEPT entries at the level corresponding to the SHARED bit, and associated with a SHARED bit value of 1, are allowed to be written.

TDH.MEM.SHARED.SEPT.WR updates the L1 SEPT tree and all L2 SEPT trees (per the number of L2 VMs configured for the TD).

To understand the table and text below, please refer to the [TDX Module Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.131: TDH.MEM.SHARED.SEPT.WR Operands Information Definition

Explicit/ Implicit	Reg.	Ref. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA and Level	L1 Secure EPT entry	SEPT_ENTRY	R	Private	$2^{12+9*Level}$ Bytes	None	None	None
Explicit	RDX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A

Explicit/ Implicit	Reg.	Ref. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	GPA	L1 Secure EPT Tree	N/A	R	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT Trees	N/A	R	Private	N/A	Shared(i)	N/A	N/A
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	None	N/A	N/A

TDH.MEM.SHARED.SEPT.WR checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must have been initialized by TDH.MNG.INIT, or an import session is in progress, started by TDH.IMPORT.STATE.IMMUTABLE and not failed.
5. The specified GPA is a valid shared GPA.
6. The specified level is the level indexed by GPA.SHARED bit (i.e., GPAW + 3).

If successful, find the SEPT entries:

7. If the specified version number is 0 or the L1 SEPT entry value provided in R8 is not NULL_PA (-1):
   7.1. Walk the L1 Secure EPT based on the GPA operand and find the L1 Secure EPT entry.
8. If the specified version number is not 0:
   8.1. For each L2 VM:
      8.1.1. If the L2 SEPT entry value provided in R9, R10 or R11 is not NULL_PA (-1), walk the VM's L2 Secure EPT based on the GPA operand and find the L2 Secure EPT entry.

If all SEPT entries were found, commit the change:

9. If the specified version number is 0 or the L1 SEPT entry value provided in R8 is not NULL_PA (-1):
   9.1. Write the value in R8 to the L1 SEPT entry.
10. If the specified version number is not 0:
   10.1. For each L2 VM:
      10.1.1. If the L2 SEPT entry value provided in R9, R10 or R11 is not NULL_PA (-1), write the value specified by the VMM to the L2 SEPT entry.

Completion Status Codes

Table 5.132: TDH.MEM.SHARED.SEPT.WR Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_WALK_FAILED	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_TDCS_NOT_ALLOCATED	
TDX_PAGE_METADATA_INCORRECT	

Completion Status Code	Description
TDX_SUCCESS	
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_INITIALIZED	

5.4.33. TDH.MEM.TRACK Leaf

Increment the TD's TLB epoch counter.

Table 5.133: TDH.MEM.TRACK Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of the parent TDR page (HKID bits must be 0)		

Table 5.134: TDH.MEM.TRACK Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.TRACK increments the TD's TLB epoch counter.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.135: TDH.MEM.TRACK Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDR	RW	Opaque	4KB	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	TDCS Epoch Tracking Fields	N/A	RW	Opaque	N/A	Exclusive	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must have been initialized (by TDH.MNG.INIT or TDH.IMPORT.STATE.IMMUTABLE).

If successful, the function does the following as a critical section, protected by exclusively locking the TDCS epoch tracking fields TD_EPOCH and REFCOUNT. A concurrent TDH.VP.ENTER may cause this locking to fail with a TDX_OPERAND_BUSY status code; in this case the caller is expected to retry TDH.MEM.TRACK.

5. Lock the TDCS epoch tracking fields in exclusive mode.
6. Check that the TD's previous epoch's REFCOUNT is 0. This helps ensure that no REFCOUNT information will be lost when TD_EPOCH is incremented in the next step.
7. If successful, increment the TD's epoch counter (TDCS.TD_EPOCH).
8. Release the exclusive mode locking of the epoch tracking fields.

Completion Status Codes

Table 5.136: TDH.MEM.TRACK Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. Note the special case where the indicated operand is TLB_EPOCH. This may happen due to a conflict with TDH.VP.ENTER. The host VMM should retry TDH.MEM.TRACK.
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_PREVIOUS_TLB_EPOCH_BUSY	
TDX_SUCCESS	TDH.MEM.TRACK is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.34. TDH.MEM.WR Leaf

Write a 64b chunk from a debuggable guest TD private memory.

Table 5.137: TDH.MEM.WR Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The guest physical address of a naturally aligned 8-byte chunk of a guest TD private page		
RDX	Host physical address of the parent TDR page (HKID bits must be 0)		
R8	Data to be written to memory		

Table 5.138: TDH.MEM.WR Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RCX	Secure EPT entry architectural content – see 3.6.2 The architectural content represents how the Secure EPT maps a private memory page or a Secure EPT page and may be different than the actual contents of the Secure EPT entry. Software should consult the Secure EPT information returned in RDX. • In case of successful operation, the requested entry's architectural content is returned. • In case of EPT walk error, the architectural content of the Secure EPT entry where the error was detected is returned. In other cases, RCX returns 0.
RDX	Secure EPT entry level and state – see 3.6.2 • In case of successful operation, the requested entry's information is returned. • In case of EPT walk error, the information of the Secure EPT entry where the error was detected is returned. In other cases, RDX returns 0.
R8	Previous content of the memory chunk In case of an error, as indicated by RAX, R8 returns 0
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.WR writes a 64b chunk to a debuggable guest TD private memory.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.139: TDH.MEM.WR Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA	TD private memory	Blob	RW	Private	8B	None	None	None
Explicit	RDX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	R	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT tree	N/A	R	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	R	Private	N/A	Exclusive	N/A	N/A

TDH.MEM.WR checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.KEY_STATE is TD_KEYS_CONFIGURED).
4. TDCS must have been initialized (TDR.INIT is TRUE).
5. The TD is debuggable (TDCS.ATTRIBUTES.DEBUG is 1).

If successful, the function does the following:

6. Walk the Secure EPT based on the GPA operand and find the leaf entry.
7. Check that the Secure EPT entry state is PRESENT.

If passed:

8. Read the content of the memory chunk.
9. Write the new content of the memory chunk.

Completion Status Codes

Table 5.140: TDH.MEM.WR Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_NOT_PRESENT	
TDX_EPT_ENTRY_STATE_INCORRECT	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	
TDX_TD_FATAL	

Completion Status Code	Description
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

Intel TDX Application Binary Interface (ABI) Reference

5.4.35. TDH.MIG.STREAM.CREATE Leaf

Create a Migration Stream and its MIGSC control structure.

Table 5.141: TDH.MIG.STREAM.CREATE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of a page where MIGSC will be created		
RDX	The physical address of the owner TDR page (HKID bits must be 0)		

Table 5.142: TDH.MIG.STREAM.CREATE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MIG.STREAM.CREATE creates a new Migration Stream and its MIGSC control structure. This function can be invoked at any time after the TDCS pages have been allocated.

TDH.MIG.STREAM.CREATE can only be successfully invoked if no migration session is in progress.

Enumeration: Availability of TDH.MIG.STREAM.CREATE is enumerated by TDX_FEATURES0.TD_MIGRATION (bit 0), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.EXPORT.ABORT returns a TDX_OPERAND_INVALID(RAX) status.

Cache Lines Flushing (Future): On future platforms, if cache line flushing is required, as enumerated by TDX_FEATURES0.CLFLUSH_BEFORE_ALLOC (bit 23), then before calling TDH.MIG.STREAM.CREATE, the host VMM should ensure that no cache lines associated with the added MIGSC physical page are in a Modified state, as described in the [Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.143: TDH.MIG.STREAM.CREATE Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	MIGSC page	MIGSC	RW	Opaque	4KB	Exclusive	Shared	Shared
Explicit	RDX	HPA	TDR page	TDR	RW	Opaque	4KB	Shared	Shared	Shared

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	4KB	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive(h)	N/A	N/A
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	Exclusive	N/A	N/A
Implicit	N/A	N/A	Mig. Stream context	Mig. Stream context	RW	Opaque	N/A	Exclusive(i)	N/A	N/A

In addition to the explicit memory operand checks per the table above, the function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. TDCS pages have been allocated (TDR.NUM_TDCX is the required number).
5. No migration session is in progress (TDCS.OP_STATE is none of *_EXPORT or *_IMPORT).
6. The MIGSC page metadata in PAMT is correct (PT is PT_NDA).
7. The number of already created migration streams is lower than the maximum allowed.

If successful, the function does the following:

8. Increment the number of migration streams (TDCS.NUM_MIG_STREAMS).
9. On platforms which use ACT, set the page's ACT bit to 1
10. Initialize the MIGSC page contents using direct write (MOVDIR64B).
11. Initialize the applicable forward link entry in TDCS (TDCS.MIGSC_LINK):
    - Set MIGSC_PA to the MIGSC page HPA.
    - Clear the INITIALIZED and ENABLED flags.
12. Atomically increment TDR.CHLDCNT.
13. Initialize the MIGSC page metadata in PAMT (Set PT to PT_TDCX, OWNER to the TDR HPA).

Completion Status Codes

Table 5.144: TDH.MIG.STREAM.CREATE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MIG.STREAM.CREATE is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	

Completion Status Code	Description
TDX_TDCS_NOT_ALLOCATED	

Intel TDX Application Binary Interface (ABI) Reference

5.4.36. TDH.MNG.ADDCX Leaf

Add a TDCS physical page to a guest TD.

Table 5.145: TDH.MNG.ADDCX Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of a page where TDCX will be added (HKID bits must be 0)		
RDX	The physical address of the owner TDR page (HKID bits must be 0)		

Table 5.146: TDH.MNG.ADDCX Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MNG.ADDCX adds a TDCS physical page, which is a child of the specified TDR.

Cache Lines Flushing (Future): On future platforms, if cache line flushing is required, as enumerated by TDX_FEATURES0.CLFLUSH_BEFORE_ALLOC (bit 23), then before calling TDH.MNG.ADDCX, the host VMM should ensure that no cache lines associated with the added TDCS physical page are in a Modified state, as described in the [Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.147: TDH.MNG.ADDCX Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[16]
Explicit	RCX	HPA	TDCX page	Blob	RW	Opaque	4KB	Exclusive	Shared	Shared	Exclusive
Explicit	RDX	HPA	TDR page	TDR	RW	Opaque	4KB	Exclusive	Shared	Shared	None

[16] ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

In addition to the explicit memory operand checks per the table above, the function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The number of TDCX pages (TDR.NUM_TDCX) is smaller than the required number.
4. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
5. The new TDCX page metadata in PAMT must be correct (PT must be PT_NDA).

If successful, the function does the following:

6. On platforms which use ACT, set the page's ACT bit to 1.
7. Initialize the TDCX page contents using direct writes (MOVDIR64B).
8. Set the TDCX pointer entry in the TDR.TDCX_PA array.
9. Increment TDR.NUM_TDCX.
10. If TDR.NUM_TDCX is equal to the required number of TDCX pages:
    10.1. Mark the TD as uninitialized (set TDCS.OP_STATE to UNINITIALIZED).
    10.2. Generate a migration encryption key, to be used in the next migration session.

    If failed:

    10.3. Decrement TDR.NUM_TDCX.

Completion Status Codes

Table 5.148: TDH.MNG.ADDCX Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_RND_NO_ENTROPY	Failed to generate a random migration encryption key. This is typically caused by an entropy error of the CPU's random number generator, and may be impacted by RDSEED, RDRAND or PCONFIG executing on other LPs. The operation should be retried.
TDX_SUCCESS	TDH.MNG.ADDCX is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCX_NUM_INCORRECT	

5.4.37. TDH.MNG.CREATE Leaf

Create a new guest TD and its TDR root page.

Table 5.149: TDH.MNG.CREATE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of a page where TDR will be created (HKID bits must be 0)		
RDX	**Bits**	**Name**	**Description**
	15:0	HKID	The TD's ephemeral private HKID
	63:16	Reserved	Reserved: must be 0

Table 5.150: TDH.MNG.CREATE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MNG.CREATE creates a TDR page which is the root page of a new guest TD.

Cache Lines Flushing (Future): On future platforms, if cache line flushing is required, as enumerated by TDX_FEATURES0.CLFLUSH_BEFORE_ALLOC (bit 23), then before calling TDH.MNG.CREATE, the host VMM should ensure that no cache lines associated with the new TDX physical page are in a Modified state, as described in the [Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.151: TDH.MNG.CREATE Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[17]
Explicit	RCX	HPA	TDR page	TDR	RW	Opaque	4KB	Exclusive	Shared	Shared	Exclusive
Implicit	N/A	N/A	KOT	KOT	N/A	Hidden	N/A	Exclusive	N/A	N/A	None

In addition to the explicit memory operand checks per the table above, the function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_NDA).
2. The value of the specified HKID must be in the range configured for TDX.
3. The KOT entry for the specified HKID must be marked as HKID_FREE.

If successful, the function does the following:

4. On platforms which use ACT, set the page's ACT bit to 1.
5. Zero out the TDR page contents using direct write (MOVDIR64B).
6. Initialize the key management fields.
7. Initialize the state variables.
8. Initialize the TD management fields.
9. Initialize the TD preserving fields (handoff version and current SEAMDB entry's index/nonce pair).
10. Mark the KOT entry for the specified HKID as HKID_ASSIGNED.
11. Initialize the TDR page metadata in PAMT.

Completion Status Codes

Table 5.152: TDH.MNG.CREATE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_HKID_NOT_FREE	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_RND_NO_ENTROPY	Random TD_UUID generation (e.g., RDRAND or RDSEED) failed because the hardware random number generator did not have enough entropy. The host VMM should retry the operation.
TDX_SUCCESS	TDH.MNG.CREATE is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	

[17] ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

5.4.38. TDH.MNG.INIT Leaf

Initialize TD-scope control structures TDR and TDCS.

Table 5.153: TDH.MNG.INIT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	TD handle and flags:		
	Bits	**Name**	**Description**
	0	EVENT_FILTERING	Flags that performance monitoring events are filtered based on the EVENT_FILTER array specified by R8. **Enumeration:** Support of this flag is enumerated by TDX_FEATURES0.EVENT_FILTERING (bit 24). If not supported, its value must be 0.
	11:1	Reserved	Reserved: must be 0
	51:12	TDR_HPA	Bits 51:12 of the host physical address of the parent TDR page (HKID bits must be 0)
	63:52	Reserved	Reserved: must be 0
RDX	The physical address (including HKID bits) of an input TD_PARAMS_STRUCT		
R8	If RCX.EVENT_FILTERING is 0 or the configured ATTRIBUTES.PERFMON is 0, then R8 is ignored. Else, R8 provides the following information:		
	Bits	**Name**	**Description**
	11:0	EVENT_FILTERS_NUM	The number of valid entries in the EVENT_FILTERS_ARRAY. Must be higher than 0 and lower or equal to MAX_EVENT_FILTERS, readable by TDH.SYS.RD*.
	51:12	EVENT_FILTERS_HPA	Bits 51:12 of the shared HPA (including HKID) of an array of EVENT_FILTER entries. The entry format and array restrictions are defined in 3.4.6.
	63:52	Reserved	Reserved: must be 0

Table 5.154: TDH.MNG.INIT Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1

Operand	Description		
RCX	Extended error information In case of a TD_PARAMS_STRUCT.CPUID_CONFIG error, RCX returns the applicable CPUID information as shown below. In all other cases, RCX returns 0.		
	Bits	**Name**	**Description**
	31:0	LEAF	CPUID leaf number
	63:32	SUBLEAF	CPUID sub-leaf number: if sub-leaf is not applicable, value is -1 (0xFFFFFFFF).
Other	Unmodified		

Leaf Function Latency

TDH.MNG.INIT execution time may be longer than most TDX module interface functions execution time. No interrupts (including NMI and SMI) are processed by the logical processor during that time.

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MNG.INIT initializes the TD-scope control structures TDR and TDCS based on a set of TD parameters provided as input.

Enumeration: Support of RCX.EVENT_FILTERING and the EVENT_FILTERS array is enumerated by TDX_FEATURES0.PERFMON_EVENT_FILTERING (bit 24), readable by TDH.SYS.RD*.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.155: TDH.MNG.INIT Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[18]
Explicit	RCX	HPA	TDR page	TDR	RW	Opaque	4KB	Exclusive	Shared	Shared	None
Explicit	RDX	HPA	TD Parameters	TD_PARAMS	R	Shared	1024B	None	N/A	N/A	None
Explicit	R8[19]	HPA	EVENT_FILTERS	EVENT_FILTER array	RW	Shared	4KB	None	N/A	N/A	None
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Exclusive(i)	N/A	N/A	None
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive(i)	N/A	N/A	None

In addition to the explicit memory operand checks per the table above, the function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).

[18] ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

[19] Only if RCX.EVENT_FILTERING is set to 1

4. All the required TDCS pages have been added (by TDH.MNG.ADDCX) but the TD has not have been initialized (TDCS.OP_STATE is UNINITIALIZED).
5. RCX.EVENT_FILTERING can only be set if supported by the TDX module.

If successful, the function does the following:

6. Set the TDCS TD management fields to their initial values.
7. Read the input parameters structure fields.
8. Check the input parameters and initialize the TDCS logical structure.
   8.1. Check that ATTRIBUTES and XFAM bits that must be fixed-0 or fixed-1 are set correctly.
   8.2. Check XFAM bit groups that must have certain values (e.g., AVX bits 7:5).
   8.3. Check the other input parameters. See the definition of TD_PARAMS in 3.4.5 for details.
   8.4. If RCX.EVENT_FILTERING is supported and is set, copy the input EVENT_FILTERS array to TDCS, while checking its validity as specified in 3.4.6.

If passed:

9. Initialize EPTP to point to TDCS.SEPT_ROOT.
10. Initialize the MSR bitmaps based on ATTRIBUTES and XFAM.
11. Initialize the TDCS measurement fields.
12. Mark the TD as initialized (set TDCS.OP_STATE to INITIALIZED).

Completion Status Codes

Table 5.156: TDH.MNG.INIT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EVENT_FILTER_INVALID	
TDX_EVENT_FILTER_ORDER_INVALID	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MNG.INIT is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	
TDX_TDCX_NUM_INCORRECT	

5.4.39. TDH.MNG.KEY.CONFIG Leaf

Configure the TD ephemeral private key on a single package.

Table 5.157: TDH.MNG.KEY.CONFIG Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of a TDR page (HKID bits must be 0)		

Table 5.158: TDH.MNG.KEY.CONFIG Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Latency

TDH.MNG.KEY.CONFIG execution time may be longer than most TDX module interface functions execution time. No interrupts (including NMI and SMI) are processed by the logical processor during that time.

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MNG.KEY.CONFIG configures the TD's ephemeral private key on a single package.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.159: TDH.MNG.KEY.CONFIG Operands Information

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	RW	Opaque	4KB	Exclusive	Shared	Shared
Implicit	N/A	N/A	KETs on current package	N/A	N/A	Hidden	N/A	Exclusive	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. HKID has been assigned to the TD; TDR.LIFECYCLE_STATE is TD_HKID_ASSIGNED.

If successful, the function does the following:

4. Configure the TD ephemeral private key on the package.
   4.1. This operation may fail due to a conflict with a concurrent TDH.MNG.KEY.CONFIG or PCONFIG running on the same package.
   4.2. A CPU-generated random key is used. The operation may fail due to lack of entropy.
5. If the key has been configured on all the packages, set TDR.LIFECYCLE_STATE to TD_KEYS_CONFIGURED.

Completion Status Codes

Table 5.160: TDH.MNG.KEY.CONFIG Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_KEY_CONFIGURED	
TDX_KEY_GENERATION_FAILED	Failed to generate a random key. This is typically caused by an entropy error of the CPU's random number generator, and may be impacted by RDSEED, RDRAND or PCONFIG executing on other LPs. The operation should be retried.
TDX_LIFECYCLE_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. Specifically, key configuration may fail due to a concurrently running PCONFIG instruction.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MNG.KEY.CONFIG is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	

5.4.40. TDH.MNG.KEY.FREEID Leaf

End the platform cache flush sequence and mark applicable HKIDs in KOT as free.

Table 5.161: TDH.MNG.KEY.FREEID Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of a TDR page (HKID bits must be 0)		

Table 5.162: TDH.MNG.KEY.FREEID Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MNG.KEY.FREEID ends the platform cache flush sequence for the HKIDs associated with the specified TD after TDH.PHYMEM.CACHE.WB has been executed (unless that function is not required, as enumerated by TDX_FEATURES0.SKIP_PHYMEM_CACHE_WB (bit 34)) on all the required WBINVD domains. It marks the TD's HKIDs in KOT as free, and the TD itself as being torn down.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.163: TDH.MNG.KEY.FREEID Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	RW	Opaque	4KB	Exclusive	Shared	Shared
Implicit	N/A	N/A	KOT	KOT	N/A	Hidden	N/A	Exclusive	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. TLB and VMCS caches associated with the HKID have been flushed, and no memory associated with this HKID may be accessed, i.e., all the following conditions are met:
   2.1. TDR.LIFECYCLE_STATE is TD_BLOCKED.

2.2. The KOT entry for the TD's private HKID is marked as HKID_FLUSHED.
2.3. If TDH.PHYMEM.CACHE.WB is required, as enumerated by TDX_FEATURES0.SKIP_PHYMEM_CACHE_WB (bit 34) value of 0, the KOT entry for the TD's private HKID indicates that TDH.PHYMEM.CACHE.WB has been executed on all applicable packages or cores.

If successful, the function does the following:

3. Mark the KOT entry as HKID_FREE.
4. Set TDR.LIFECYCLE_STATE to TD_TEARDOWN.

Completion Status Codes

Table 5.164: TDH.MNG.KEY.FREEID Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_LIFECYCLE_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MNG.KEY.FREEID is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_WBCACHE_NOT_COMPLETE	

5.4.41. TDH.MNG.KEY.RECLAIMID Leaf (Deprecated)

This function is deprecated; it is provided for backward compatibility.

Table 5.165: TDH.MNG.KEY.RECLAIMID Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0

Table 5.166: TDH.MNG.KEY.RECLAIMID Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MNG.KEY.RECLAIMID is provided for backward compatibility. It does not do anything except returning a constant TDX_SUCCESS status.

Completion Status Codes

Table 5.167: TDH.MNG.KEY.RECLAIMID Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_SUCCESS	TDH.MNG.KEY.RECLAIMID is successful.

5.4.42. TDH.MNG.RD Leaf

Read a TD-scope metadata field (control structure field) of a TD.

Table 5.168: TDH.MNG.RD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Version number may be 0 or 1. See the enumeration details below.
	63:24	Reserved	Must be 0
RCX	The physical address of a TDR page (HKID bits must be 0)		
RDX	Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored. For TDH.MNG.RD version 1 or higher, a value of -1 is a special case: it is not a valid field identifier; in this case the first readable field identifier is returned in RDX.		

Table 5.169: TDH.MNG.RD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RDX	For TDH.MNG.RD version 0, RDX is unmodified. For TDH.MNG.RD version 1 or higher: • If the input field identifier was -1, RDX returns the first readable field identifier. Else, in case of an error, RDX returns -1. On success, RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available.
R8	Contents of the field In case of no success, as indicated by RAX, R8 returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MNG.RD reads a TD-scope metadata field (control structure field) of a TD.

Enumeration: Availability of TDH.MNG.RD version 1 is enumerated by TDX_FEATURES0.ENHANCED_METADATA (bit 3), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.MNG.RD with a version number higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

If version 1 or higher is specified in RAX, RDX returns the next host-side readable field identifier. This may be used by the host VMM to dump the host readable TD metadata. To read all the available fields, the host VMM can invoke

TDH.MNG.RD in a loop, starting with field identifier -1 as an input, until RDX returns -1. A status code of TDX_METADATA_FIELD_SKIP indicates that the returned value is not applicable.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.170: TDH.MNG.RD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. All the required TDCS pages have been added (TDR.NUM_TDCX is the required number).

If the above checks passed:

5. Read the control structure field using the algorithm described in 5.3.2.1.

Completion Status Codes

Table 5.171: TDH.MNG.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_METADATA_FIELD_ID_INCORRECT	
TDX_METADATA_FIELD_NOT_READABLE	
TDX_METADATA_FIELD_SKIP	Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any.
TDX_METADATA_FIRST_FIELD_ID_IN_CONTEXT	Indicates that the first field ID in context is returned
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MNG.RD is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	

Completion Status Code	Description
TDX_TDCS_NOT_ALLOCATED	

5.4.43. TDH.MNG.VPFLUSHDONE Leaf

Check that none of the TD's VCPUs are associated with an LP.

Table 5.172: TDH.MNG.VPFLUSHDONE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of a TDR page (HKID bits must be 0)		

Table 5.173: TDH.MNG.VPFLUSHDONE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MNG.VPFLUSHDONE checks that none of the TD's VCPUs are associated with an LP, and it then prepares for cache flushing by TDH.PHYMEM.CACHE.WB.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.174: TDH.MNG.VPFLUSHDONE Operands Information

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	RW	Opaque	4KB	Exclusive	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	KOT	KOT	N/A	Hidden	N/A	Exclusive	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	None	Opaque	N/A	Exclusive(i)	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. TDR.LIFECYCLE_STATE is either TD_HKID_ASSIGNED or TD_KEYS_CONFIGURED.
3. The KOT entry for the TD's assigned HKID in the list must be marked as HKID_ASSIGNED.
4. None of the TD's VCPUs are associated with an LP (either the TD has not been initialized by TDH.MNG.INIT, or TDCS.NUM_ASSOC_VCPUS is 0).

If successful, the function does the following:

5. Set a bitmap in the KOT entry to track the required subsequent TDH.PHYMEM.CACHE.WB operations.
6. Set TDR.LIFECYCLE_STATE to TD_BLOCKED.
7. Mark the KOT entry as HKID_FLUSHED.

Completion Status Codes

Table 5.175: TDH.MNG.VPFLUSHDONE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_FLUSHVP_NOT_DONE	
TDX_IOMMU_IOTLB_TRACKING_NOT_DONE	Applicable only if the TDX module supports TDX Connect.
TDX_LIFECYCLE_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MNG.VPFLUSHDONE is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_HAS_ATTACHED_DEVICES	Applicable only if the TDX module supports TDX Connect.

5.4.44. TDH.MNG.WR Leaf

Write a TD-scope metadata field (control structure field) of a TD.

Table 5.176: TDH.MNG.WR Input Operands Definitions

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of a TDR page (HKID bits must be 0)		
RDX	Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored.		
R8	Data to write to the field		
R9	A 64b write mask to indicate which bits of the value in R8 are to be written to the field		

Table 5.177: TDH.MNG.WR Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
R8	Previous content of the field In case of an error, as indicated by RAX, R8 returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MNG.WR writes a TD-scope metadata field (control structure field) of a TD. The value (R8) is written as specified by the write mask (R9). Writing is subject to the field's internal write mask (per the TD's ATTRIBUTES.DEBUG bit). Writing of specific fields is also subject to additional rules.

Table 5.178: Metadata Field Write Rules

Write Mask Bit in R9	Internal Write Mask Bit	Value Bit in R8
0	N/A	Silently ignored
1	0	Must be the same as the current field's bit
1	1	Written to the current field's bit

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.179: TDH.MNG.WR Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	RW	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR)
2. The TD is not in a FATAL state (TDR.FATAL is FALSE)
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED)
4. All the required TDCS pages have been added (TDR.NUM_TDCX is the required number).

If the above checks passed:

5. Write the control structure field and return its old value, using the algorithm described in 5.3.2.2.

Completion Status Codes

Table 5.180: TDH.MNG.WR Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_METADATA_FIELD_ID_INCORRECT	
TDX_METADATA_FIELD_NOT_READABLE	
TDX_METADATA_FIELD_NOT_WRITABLE	
TDX_METADATA_FIELD_VALUE_NOT_VALID	
TDX_METADATA_WR_MASK_NOT_VALID	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MNG.WR is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NON_DEBUG	
TDX_TDCS_NOT_ALLOCATED	

5.4.45. TDH.MR.EXTEND Leaf

Extend the MRTD measurement register in the TDCS with the measurement of the indicated chunk of a TD page.

Table 5.181: TDH.MR.EXTEND Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The GPA of the TD page chunk to be measured		
RDX	The physical address of the TDR page of the target TD (HKID bits must be 0)		

Table 5.182: TDH.MR.EXTEND Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry where the error was detected In other cases, RCX returns 0.
RDX	Extended error information part 2 In case of EPT walk error, EPT level where the error was detected In other cases, RDX returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MR.EXTEND updates the MRTD measurement register in the TDCS with the measurement of the indicated chunk of a TD private page. For pages whose contents need to be measured, once the page is copied into the TD memory area, the host VMM will call TDH.MR.EXTEND multiple times to measure the pages contents into MRTD. TDEXEND can be executed only before TDH.MR.FINALIZE.

Note: TDH.MR.EXTEND works on a 256B chunk of a page, not on a full page, due to instruction latency considerations.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.183: TDH.MR.EXTEND Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA	TD private page chunk	Blob	R	Private	256B	None	None	None
Explicit	RDX	HPA	TDR page	TDR	R	Opaque	4KB	Exclusive	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	4KB	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	GPA	Secure EPT tree	N/A	R	Private	N/A	Exclusive(i)	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive(i)	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must have been initialized but not finalized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is INITIALIZED).
5. The page must be mapped and accessible in the Secure EPT.

If successful, the function does the following:

6. Update the TD measurement in TDCS based on the chunk's GPA and contents.
7. Extend TDCS.MRTD with the chunk's GPA and contents. Extension is done using SHA384, with three 128B extension buffers. The first extension buffer is composed as follows:
   - Bytes 0 through 8 contain the ASCII string "MR.EXTEND".
   - Bytes 16 through 23 contain the GPA (in little-endian format).
   - All the other bytes contain 0.

   The other two extension buffers contain the chunk's contents.

Completion Status Codes

Table 5.184: TDH.MR.EXTEND Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_NOT_PRESENT	
TDX_EPT_WALK_FAILED	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MR.EXTEND is successful.
TDX_SYS_NOT_READY	

Completion Status Code	Description
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.46. TDH.MR.FINALIZE Leaf

TDH.MR.FINALIZE completes measurement of the initial TD contents and marks the TD as ready to run.

Table 5.185: TDH.MR.FINALIZE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of the parent TDR page (HKID bits must be 0)		

Table 5.186: TDH.MR.FINALIZE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MR.FINALIZE completes the measurement of the initial TD contents and marks the TD as finalized.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.187: TDH.MR.FINALIZE Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDR page	TDR	R	Opaque	4KB	Exclusive	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	4KB	Exclusive(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive(i)	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must have been initialized but not finalized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is INITIALIZED).

If successful, the function does the following:

5. Finalize the TD measurement, i.e., SHA384 calculation of TDCS.MRTD that has been accumulated so far by TDH.MEM.PAGE.ADD and TDH.MR.EXTEND.
6. Calculate TDCS.SERVTD_HASH:
    6.1. Get all service TD binding slots whose SERVTD_BINDING_STATE is not NOT_BOUND.
        6.1.1. If no service TD binding slots apply, set TDCS.SERVTD_HASH to 0.
    6.2. Sort in ascending order by SERVTD_TYPE as the primary key, SERVTD_INFO_HASH as a secondary key (if multiple service TDs of the same type are bound).
    6.3. Concatenate SERVTD_INFO_HASH, SERVTD_TYPE and SERVTD_ATTR of each slot in a temporary buffer:
        6.3.1. SERVTD_INFO_HASH in bytes 5:0
        6.3.2. SERVTD_TYPE in bytes 7:6
        6.3.3. SERVTD_ATTR in bytes 15:8
        6.3.4. Concatenate all buffers.
        6.3.5. Calculate SHA384 and store in TDCS.SERVTD_HASH.
7. Mark the TD as finalized.

Completion Status Codes

Table 5.188: TDH.MR.FINALIZE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MR.FINALIZE is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.47. TDH.PHYMEM.CACHE.WB Leaf

TDH.PHYMEM.CACHE.WB is an interruptible and resumable function to write back the cache hierarchy on a package or a core.

If the value of TDX_FEATURES0.SKIP_PHYMEM_CACHE_WB (bit 34), readable by TDH.SYS.RD*, is 1, TDH.PHYMEM.CACHE.WB is provided for backward compatibility. It returns immediately, indicating success.

Table 5.189: TDH.PHYMEM.CACHE.WB Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	Command, as described below:		
	Value	**Name**	**Description**
	0	WB_START_CMD	Start a new TDH.PHYMEM.CACHE.WB cycle with no cache invalidation.
	1	WB_RESUME_CMD	Resume a previously interrupted TDH.PHYMEM.CACHE.WB cycle with no cache invalidation.
	Other		Reserved

Table 5.190: TDH.PHYMEM.CACHE.WB Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.PHYMEM.CACHE.WB writes back the cache hierarchy to memory and updates the KOT state to allow reuse of HKIDs.

Enumeration: If the value of TDX_FEATURES0.SKIP_PHYMEM_CACHE_WB (bit 34), readable by TDH.SYS.RD*, is 1, TDH.PHYMEM.CACHE.WB is provided for backward compatibility. It does nothing and returns immediately with TDX_SUCCESS.

Interruptibility: TDH.PHYMEM.CACHE.WB is interruptible. If a pending interrupt is detected during operation, TDH.PHYMEM.CACHE.WB returns with a TDX_INTERRUPTED_RESUMABLE status in RAX.

The hosts VMM initially calls TDH.PHYMEM.CACHE.WB with RCX indicating WB_START_CMD. If TDH.PHYMEM.CACHE.WB returns TDX_INTERRUPTED_RESUMABLE (or any other recoverable error) status in RAX, the host VMM should call TDH.PHYMEM.CACHE.WB with RCX indicating WB_RESUME_CMD in a loop until it completes its operation, as indicated by TDX_SUCCESS status.

Warning: When TDH.PHYMEM.CACHE.WB is interrupted, the CPU still considers this as a cache write-back operation in progress. The host VMM should complete the cache write-back operation by resuming

TDH.PHYMEM.CACHE.WB (i.e., with RCX indicating WB_RESUME_CMD) until completed successfully. **Failure to do so will result in memory performance impact that would only be resolved by a restart.** The host VMM should also refrain from executing WBINVD or WBNOINVD while the TDH.PHYMEM.CACHE.WB cycle is in progress.

Other TDH.PHYMEM.CACHE.WB characteristics:

- TDH.PHYMEM.CACHE.WB does not invalidate cache lines.
- The function operates on cache lines associated with any HKID.
- The function is designed to ensure write back of at least those cache lines where the state of that HKID (in the KOT) was HKID_FLUSHED at the time of the first invocation (RCX == WB_START_CMD).
- Depending on the implementation, the instruction may write back additional cache lines.
- The scope at which TDH.PHYMEM.CACHE.WB operates (e.g., package or core) is determined at Intel TDX module initialization time.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.191: TDH.PHYMEM.CACHE.WB (Implicit) Operands Information

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	N/A	KOT	KOT	N/A	Hidden	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	WBT entry for current scope	WBT_ENTRY	N/A	Hidden	N/A	Exclusive	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. The command value is one of the supported ones.
2. If the command is to start a new TDH.PHYMEM.CACHE.WB cycle (RCX == 0), then:
   2.1. Clear the internally saved interruption state.
   2.2. Scan the KOT: mark those HKIDs whose state is HKID_FLUSHED in an internal table; only those HKIDs will be later marked as written back upon successful completion of TDH.PHYMEM.CACHE.WB.
   2.3. If none of the KOT entries for the requested set of HKIDs (either single or all) is in HKID_FLUSHED state, then abort with an informational code (it achieved its goal: write back and invalidate at least the HKIDs that are in the HKID_FLUSHED state).
3. Run cache write back operation on the cache hierarchy of the current WBINVD domain. This operation is long and may be interrupted by external events.
   3.1. If a previous TDH.PHYMEM.CACHE.WB has been interrupted, the operation resumes from the interruption point which has been recorded.
   3.2. In case of interruption, the current point in the write back and invalidation flow and the current HKID are recorded.
4. If the operation has not been interrupted, update the KOT as follows:
   4.1. For each KOT entry, if the entry was marked as HKID_FLUSHED at the start of the TDH.PHYMEM.CACHE.WB cycle as discussed above, use the KOT entry's bitmap to indicate that TDH.PHYMEM.CACHE.WB has been executed on this package or core.

Error and Informational Codes

Table 5.192: TDH.PHYMEM.CACHE.WB Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	TDH.PHYMEM.CACHE.WB was interrupted; it is recommended to resume it with RCX indicating WB_RESUME_CMD
TDX_NO_HKID_READY_TO_WBCACHE	

Completion Status Code	Description
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDH.PHYMEM.CACHE.WB is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	

5.4.48. TDH.PHYMEM.PAGE.RDMD Leaf

Read the metadata of a page (or the metadata of the containing large page) in TDMR.

Table 5.193: TDH.PHYMEM.PAGE.RDMD Operands

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	A physical address of a 4KB page in TDMR (HKID bits must be 0)		

Table 5.194: TDH.PHYMEM.PAGE.RDMD Output Operands Definition

Operand	Description		
RAX	SEAMCALL instruction return code – see 5.4.1		
RCX	Page Type (PT) – see 3.5.1 In case of an error, RCX returns 0.		
RDX	For most PT values, this field returns the HPA of the TD's TDR control structure page, if applicable (HKID bits are set to 0). For PT value of PT_IOMMU_MT (applicable only if the TDX module supports TDX Connect), this field returns the IOMMU_ID. See the [TDX Connect ABI]. In multiple error cases, as indicated by RAX, RDX returns 0. In other error cases, RDX still returns the OWNER information. See the completion status codes table below for details.		
R8	**Bits**	**Name**	**Description**
	2:0	Size	Size of the containing 4KB, 2MB or 1GB page
	63:3	Reserved	Set to 0
	In case of an error, as indicated by RAX, R8 returns 0.		
R9	BEPOCH In case of an error, as indicated by RAX, R9 returns 0.		
R10	Reserved: set to 0		
R11	Reserved: set to 0		
Other	Unmodified		

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.PHYMEM.PAGE.RDMD finds the containing page (4KB, 2MB or 2GB) of the given page in TDMR and reads its metadata from its PAMT entry.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.195: TDH.PHYMEM.PAGE.RDMD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	Target page	Blob	None	Opaque/ Private	4KB	Shared	Shared	Shared

If the memory operand checks, per the table above, pass, the function does the following:

1. Do a PAMT walk and find the containing page and its size.

If passed:

2. Read the PAMT entry.

Completion Status Codes

Table 5.196: TDH.PHYMEM.PAGE.RDMD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDH.PHYMEM.PAGE.RDMD is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	

5.4.49. TDH.PHYMEM.PAGE.RECLAIM Leaf

Reclaim a physical 4KB, 2MB or 1GB TD-owned page (i.e., TD private page, Secure EPT page or a control structure page) from a TD, given its HPA.

Table 5.197: TDH.PHYMEM.PAGE.RECLAIM Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	SEAMCALL instruction leaf number and version, see 5.4.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the SEAMCALL interface function
		23:16	Version Number	Selects the SEAMCALL interface function version Must be 0 or 1, see enumeration details below.
		63:24	Reserved	Must be 0
RCX	PAGE	The physical address of a 4KB, 2MB or 1GB page to be reclaimed (HKID bits must be 0)		

5

Table 5.198: TDH.PHYMEM.PAGE.RECLAIM Output Operands Definition

Operand	Name	Description		
RAX	STATUS	SEAMCALL instruction return code – see 5.4.1		
RCX	PT	Page Type (PT) – see 3.5.1 In multiple error cases, as indicated by RAX, RCX returns 0. In other error cases, RCX still returns the PT information. See the completion status codes table below for details.		
RDX	OWNER	For most PT values, this field returns the HPA of the TD's TDR control structure page, if applicable (HKID bits are set to 0). For PT value of PT_IOMMU_MT (applicable only if the TDX module supports TDX Connect), this field returns the IOMMU_ID. See the [TDX Connect ABI]. In multiple error cases, as indicated by RAX, RDX returns 0. In other error cases, RDX still returns the OWNER information. See the completion status codes table below for details.		
R8	SIZE	**Bits**	**Name**	**Description**
		2:0	Size	Size of the containing 4KB, 2MB or 1GB page – see 3.5.1
		63:3	Reserved	Set to 0
		In multiple error cases, as indicated by RAX, RDX returns 0. In other error cases, RDX still returns the size information. See the completion status codes table below for details.		
R9		Reserved: set to 0		
R10		Reserved: set to 0		
R11		Reserved: set to 0		
Other		Unmodified		

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.PHYMEM.PAGE.RECLAIM reclaims a TD-owned physical page from the TD.

Enumeration: Support of TDH.PHYMEM.PAGE.RECLAIM version 1 is enumerated by TDX_FEATURES0.ACT (bit 14), readable by the host VMM using TDH.SYS.RD*.

Support of reclaiming PT_TR pages is enumerated by TDX_FEATURES0.NON_BLOCKING_RESIZE (bit 35).

Owner TD Lifecycle State: Normally, TDH.PHYMEM.PAGE.RECLAIM can reclaim pages only if the owner TD is in the TD_TEARDOWN state. However, if reclaiming PT_TR pages is supported, a PT_TR page may be reclaimed while the TD is in its normal operating state (TD_KEYS_CONFIGURED). In that case, if the TD may be running, the function checks the TLB tracking of the reclaimed PT_TR page.

Reclaimed Page Initialization: After the page has been reclaimed, the host VMM should initialize its content before it is reused as a non-private page, as described in the [Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.199: TDH.PHYMEM.PAGE.RECLAIM Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[20]
Explicit	RCX	HPA	Target page	Blob	RW	Opaque/ Private	4KB, 2MB or 1GB	Exclusive	Shared	Shared	Exclusive
Implicit	N/A	N/A	TDR page[21]	TDR	RW	Opaque	4KB	Shared	N/A	N/A	None
Implicit	N/A	N/A	TDCS structure[22]	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A	None
Implicit	N/A	N/A	TDCS.OP_STATE[22]	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A	None

TDH.PHYMEM.PAGE.RECLAIM checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

The function works as follows:

1. Check that the target page metadata in PAMT are correct (PT must not be PT_NDA nor PT_RSVD).
2. If the target page is not a TDR (PT is not PT_TDR):
   2.1. Get the TDR page (pointed by the target page's PAMT.OWNER).
   2.2. If reclaiming PT_TR pages is supported, as enumerated by TDX_FEATURES0.NON_BLOCKING_RESIZE (bit 35), check if all the following conditions are met:
      2.2.1. PAMT.PT is PT_TR.
      2.2.2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
      2.2.3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
      2.2.4. The TD must have been initialized or its metadata has been imported (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is either INITIALIZED, RUNNING, *_EXPORT, POST_IMPORT or LIVE_IMPORT).

[20] ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

[21] Except when TDR is the target page

[22] Only if the TDX module supports NON_BLOCKING_RESIZE, and TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED

If all conditions are met:

2.2.5. If the TD may run (its OP_STATE in either RUNNING, LIVE_EXPORT or LIVE_IMPORT), check that TLB tracking has been done, based on the page's PAMT.BEPOCH. If failed, abort with a TDX_TLB_TRACKING_NOT_DONE status.

2.3. Else, check that the TD is in teardown state (TDR.LIFECYCLE_STATE is TD_TEARDOWN).

If passed:

2.4. On platforms which use ACT, overwrite the page content with the TD's random overwrite value.
  2.4.1. If the page size is 4KB, overwrite the page with the TD's random overwrite value using MOVDIR64B, using the TD's HKID.
  2.4.2. Else, the operation is done 4KB at a time, as follows:
    2.4.2.1. If this is the first 4KB, as indicated by PAMT.PT != PT_PR:
      2.4.2.1.1. Set the page's PAMT.PT to PT_PR.
      2.4.2.1.2. Set the page's PAMT.BEPOCH to 0.
    2.4.2.2. Overwrite the current 4KB with the TD's random overwrite value using MOVDIR64B, using the TD's HKID. Start from the offset value stored in PAMT.BEPOCH.
    2.4.2.3. If this is not the last 4KB block and there is a pending interrupt, then:
      2.4.2.3.1. Save the last overwrite offset into PAMT.BEPOCH.
      2.4.2.3.2. Execute SFENCE.
      2.4.2.3.3. If the requested version number is 0, terminate without modifying any of the host VMM's CPU state from its value before SEAMCALL. The host VMM will typically re-execute the SEAMCALL after handling the interrupt.
      2.4.2.3.4. Else, terminate with TDX_INTERRUPTED_RESUMABLE status.
  2.5. Atomically decrement TDR.CHLDCNT by 1, 512 or 512^2 depending on the removed page size (4KB, 2MB or 1GB, respectively).

3. Else (target page is a TDR):
  3.1. Check that the TD is in teardown state (TDR.LIFECYCLE_STATE is TD_TEARDOWN).
  3.2. Check that TDR.CHLDCNT is 0.
  3.3. On platforms which use ACT, overwrite the TDR page content with the TD's random overwrite value, using the TDX module's HKID.

If passed:

4. On platforms which use ACT, set the page's ACT bit(s) to 0. Note that the operation depends on the page size (4KB, 2MB or 1GB).
5. Update the PAMT entry of the reclaimed page to PT_NDA.
6. Return the page metadata (as they were before PAMT update above).

Completion Status Codes

Table 5.200: TDH.PHYMEM.PAGE.RECLAIM Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_IOMMU_MT_PAGE_IN_USE	Applies only if the TDX module supports TDX Connect
TDX_LIFECYCLE_STATE_INCORRECT	RCX, RDX and R8 return the actual PT, OWNER and SIZE information.
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. If the page is not a TDR page but the owner TDR is busy, then RCX, RDX and R8 return the actual PT, OWNER and SIZE information.
TDX_OPERAND_INVALID	If the page physical address is not aligned on its size, then RCX, RDX and R8 return the actual PT, OWNER and SIZE information.

Completion Status Code	Description
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.PHYMEM.PAGE.RECLAIM is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_ASSOCIATED_PAGES_EXIST	RCX, RDX and R8 return the actual PT, OWNER and SIZE information.
TDX_TLB_TRACKING_NOT_DONE	Applicable only if reclaiming PT_TR pages, enumerated by TDX_FEATURES0.NON_BLOCKING_RESIZE (bit 35), is supported.

5.4.50. TDH.PHYMEM.PAGE.WBINVD Leaf

Write back and invalidate all cache lines associated with the specified memory page and HKID.

Table 5.201: TDH.PHYMEM.PAGE.WBINVD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	Physical address (including HKID bits) of a 4KB page in TDMR		

Table 5.202: TDH.PHYMEM.PAGE.WBINVD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.PHYMEM.PAGE.WBINVD performs cache write back and invalidation on all the cache lines associated with the specified page and HKID. The page must not be in use by the Intel TDX module (i.e., not assigned to a TD as a private page or a Secure EPT page), nor used as a control structure page.

It is the responsibility of the host VMM to track which HKID is associated with the target page; the function does not check it.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.203: TDH.PHYMEM.PAGE.WBINVD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	Target page	Blob	R	Private/ Opaque	4KB	Shared	Shared	Shared

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. The target page must be marked in PAMT as not controlled by the Intel TDX module (PT must be PT_NDA).

If successful, the function performs the following:

2. Write back and invalidate all the cache lines for the given target HPA and HKID.

Completion Status Codes

Table 5.204: TDH.PHYMEM.PAGE.WBINVD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.PHYMEM.PAGE.WBINVD is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	

Intel TDX Application Binary Interface (ABI) Reference

5.4.51. TDH.SERVTD.BIND Leaf

Bind a service TD to a target TD.

Table 5.205: TDH.SERVTD.BIND Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of the target TD's TDR page (HKID bits must be 0)		
RDX	The physical address of the service TD's TDR page (HKID bits must be 0)		
R8	Index (slot number) in the target TD's service TD binding table		
R9	SERVTD_TYPE: Service TD type		
R10	SERVTD_ATTR: Service TD attributes		

Table 5.206: TDH.SERVTD.BIND Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RCX	Binding handle In case of an error, as indicated by RAX, RCX returns 0.
R10	TD_UUID bits 63:0 In case of an error, as indicated by RAX, R10 returns 0.
R11	TD_UUID bits 127:64 In case of an error, as indicated by RAX, R11 returns 0.
R12	TD_UUID bits 191:128 In case of an error, as indicated by RAX, R12 returns 0.
R13	TD_UUID bits 255:192 In case of an error, as indicated by RAX, R13 returns 0.
AVX, AVX2 and AVX512 state	May be reset to the architectural RESET state
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.SERVTD.BIND binds a service TD to a target TD.

Enumeration: Availability of TDH.SERVTD.BIND is enumerated by TDX_FEATURES0.SERVICE_TD (bit 2), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SERVTD.BIND returns a TDX_OPERAND_INVALID(RAX) status.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.207: TDH.SERVTD.BIND Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	Target TD's TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Explicit	RDX	HPA	Service TD's TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	N/A	Target TD's TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	Target TD's TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	Binding table		RW	Opaque	N/A	Exclusive	None	None
Implicit	N/A	N/A	Service TD's TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	Service TD's TDCS.OP_STATE	OP_STATE	R	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	Service TD's TDCS.RTMR	SHA384_ HASH	N/A	Opaque	N/A	Shared	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. Target TD checks:
   - 1.1. The target TD's TDR page metadata in PAMT must be correct (PT must be PT_TDR).
   - 1.2. The target TD is not in a FATAL state (TDR.FATAL is FALSE).
   - 1.3. The target TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
   - 1.4. The target TD's TDCS pages must have been allocated (TDR.NUM_TDCX is the required number).
   - 1.5. The target TD has not been paused for export and is not in the in-order import phase.
2. Service TD checks:
   - 2.1. The service TD's TDR HPA must be different than the target TD's TDR HPA
   - 2.2. The service TD's TDR page metadata in PAMT must be correct (PT must be PT_TDR).
   - 2.3. The service TD is not in a FATAL state (TDR.FATAL is FALSE).
   - 2.4. The service TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
   - 2.5. The service TD's TDCS pages must have been allocated (TDR.NUM_TDCX is the required number).
   - 2.6. Either the service TD's measurements have been finalized (by TDH.MR.FINALIZE) or it is being imported and import is in the out-of-order phase.
3. Binding slot number does not exceed the number of available slots.
4. SERVTD_TYPE is supported.
5. If only one service TD binding instance is supported by SERVTD_TYPE, no other binding slot whose BINDIND_STATE is not NOT_BOUND may have the same SERVTD_TYPE.
6. SERVTD_ATTR is supported.

If the above checks passed:

7. If the binding slot's SERVTD_BINDING_STATE is NOT_BOUND (i.e., this is an **initial binding**):
   7.1. Check that the target TD's measurements have not been finalized (by TDH.MR.FINALIZE).
   7.2. Copy the provided SERVTD_TYPE and SERVTD_ATTR to the binding slot.
   7.3. Calculate the service TD's SERVTD_INFO_HASH and write to the binding slot's SERVTD_INFO_HASH.
   7.4. Copy the SERVTD's TD_UUID to the binding slot's SERVTD_UUID.
8. Else, if the binding slot's SERVTD_BINDING_STATE is PRE_BOUND (i.e., this is a **late initial binding**):
   8.1. Check that the requested SERVTD_TYPE is equal to the binding slot's SERVTD_TYPE.
   8.2. Check that the requested SERVTD_ATTR is equal to the binding slot's SERVTD_ATTR.
   8.3. Calculate the service TD's SERVTD_INFO_HASH and check that it is equal to the binding slot's SERVTD_INFO_HASH.
   8.4. Copy the SERVTD's TD_UUID to the binding slot's SERVTD_UUID.
9. Else, if the binding slot's SERVTD_BINDING_STATE is BOUND (i.e., this is a **rebinding**):
   9.1. Check that the requested SERVTD_TYPE is equal to the binding slot's SERVTD_TYPE.
   9.2. Check that the requested SERVTD_ATTR is equal to the binding slot's SERVTD_ATTR.
   9.3. Calculate the service TD's SERVTD_INFO_HASH.
   9.4. Check that the service TD's SERVTD_INFO_HASH is equal to the binding slot's SERVTD_INFO_HASH.
   9.5. Copy the SERVTD's TD_UUID to the binding slot's SERVTD_UUID.

If passed:

10. Set the binding slot's SERVTD_BINDING_STATE to BOUND.
11. Calculate and return the binding handle.

Completion Status Codes

Table 5.208: TDH.SERVTD.BIND Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SERVTD_ATTR_MISMATCH	
TDX_SERVTD_INFO_HASH_MISMATCH	
TDX_SERVTD_TYPE_MISMATCH	
TDX_SERVTD_UUID_MISMATCH	
TDX_SUCCESS	TDH.SERVTD.BIND is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	

5.4.52. TDH.SERVTD.PREBIND Leaf

Pre-bind a service TD to a target TD.

Table 5.209: TDH.SERVTD.PREBIND Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of the target TD's TDR page (HKID bits must be 0)		
RDX	The physical address (including HKID bits) of SERVTD_INFO_HASH, the expected SHA384 hash of the service TD's TDINFO_STRUCT		
R8	Index (slot number) in the target TD's service TD binding table		
R9	SERVTD_TYPE: Expected service TD type		
R10	SERVTD_ATTR: Expected service TD attributes		

Table 5.210: TDH.SERVTD.PREBIND Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.SERVTD.PREBIND pre-binds a service TD to a target TD, by setting its expected binding parameters.

Enumeration: Availability of TDH.SERVTD.PREBIND is enumerated by TDX_FEATURES0.SERVICE_TD (bit 2), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SERVTD.PREBIND returns a TDX_OPERAND_INVALID(RAX) status.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.211: TDH.SERVTD.PREBIND Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	Target TD's TDR page	TDR	R	Opaque	4KB	Shared	Shared	Shared

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RDX	HPA	SERVTD_INFO_ HASH	SHA384_ HASH	R	Shared	64B	N/A	N/A	N/A
Implicit	N/A	N/A	Target TD's TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	Target TD's TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	Binding table		RW	Opaque	N/A	Exclusive	None	None

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. The target TD's TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The target TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The target TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The target TD's TDCS pages must have been allocated (TDR.NUM_TDCX is the required number).
5. The target TD's measurements have not been finalized (by TDH.MR.FINALIZE).
6. Binding slot number does not exceed the number of available slots.
7. SERVTD_TYPE is supported.
8. If only one service TD binding instance is supported by SERVTD_TYPE, no other binding slot whose BINDIND_STATE is not NOT_BOUND may have the same SERVTD_TYPE.
9. SERVTD_ATTR is supported.
10. The binding slot's SERVTD_BINDING_STATE is either NOT_BOUND or PRE_BOUND.

If the above checks passed:

11. Copy the provided SERVTD_TYPE, SERVTD_ATTR and SERVTD_INFO_HASH to the binding slot.
12. Set the binding slot's SERVTD_BINDING_STATE to PRE_BOUND.

Completion Status Codes

Table 5.212: TDH.SERVTD.PREBIND Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SERVTD_ALREADY_BOUND_FOR_TYPE	
TDX_SUCCESS	TDH.SERVTD.PREBIND is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	

Completion Status Code	Description
TDX_TDCS_NOT_ALLOCATED	

5.4.53. TDH.SYS.CONFIG Leaf

Globally configure the Intel TDX module.

Table 5.213: TDH.SYS.CONFIG Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of an array of pointers, each containing the physical address of a single TDMR_INFO entry (see 3.3.3). The pointer array must be sorted such that TDMR base addresses (TDMR_INFO.TDMR_BASE) are sorted from the lowest to the highest base address, and TDMRs do not overlap with each other.		
RDX	The number of pointers in the above buffer, between 1 and 64		
R8	**Bits**	**Name**	**Description**
	15:0	HKID	Intel TDX global private HKID value
	63:16	Reserved	Reserved: must be 0

Table 5.214: TDH.SYS.CONFIG Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.SYS.CONFIG performs global (platform-scope) configuration of the Intel TDX module. This function is intended to be executed during OS/VMM boot, and thus it has relaxed latency requirements.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.215: TDH.SYS.CONFIG Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDMR Info Pointers	Array of HPA	R	Shared	512B	None	N/A	N/A
Explicit	N/A	HPA	TDMR Info	TDMR_INFO	R	Shared	512B	None	N/A	N/A

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	N/A	All Intel TDX module internal variables	N/A	RW	Hidden	N/A	Exclusive	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. Global and LP-scope initialization has been done:
   1.1. PL.SYS_STATE is SYSINIT_DONE.
   1.2. TDH.SYS.LP.INIT has been executed on all LPs.
2. The number of TDMR_INFO entries is at least 1 and does not exceed the supported number of TDMRs.
3. Check each physical address of to TDMR_INFO; read the applicable TDMR_INFO entry; check and update the internal TDMR_TABLE with TDMR, reserved areas and PAMT setup. The order of checks is not required to be exactly the same as described below.
   - TDMRs must be sorted in an ascending base address order.
   - For each TDMR:
     - TDMR base address must be aligned on 1GB.
     - TDMR size must be greater than 0 and a whole multiple of 1GB.
     - Any address within the TDMR must comply with the platform's maximum PA, and its HKID bits must be 0.
     - For each PAMT region (1G, 2M and 4K) of each TDMR:
       - PAMT base address must comply with the alignment requirements.
       - Any address within the PAMT range must comply with the platform's maximum PA, and its HKID bits must be 0.
       - The size of each PAMT region must be large enough to contain the PAMT for its associated TDMR.
     - Reserved areas within TDMR must be sorted in an ascending offset order.
     - A null reserved area (indicated by a size of 0) may be followed only by other null reserved areas.
     - For each reserved area within TDMR:
       - Offset and size must comply with the alignment and granularity requirements.
       - Reserved areas must not overlap.
       - Reserved areas must be fully contained within their TDMR.
   - TDMRs must not overlap with other TDMRs.
   - PAMTs must not overlap with other PAMTs.
   - TDMRs' non-reserved parts and PAMTs must not overlap (PAMTs may reside within TDMR reserved areas).
   - TDMRs' non-reserved parts must be contained in convertible memory – i.e., in CMRs.
   - PAMTs must be contained in convertible memory – i.e., in CMRs.
4. Check and set the Intel TDX global private HKID. The provided HKID must be in the TDX HKID range.

If successful, the function does the following:

5. Complete the initialization of the Intel TDX module at platform scope.
6. Set PL.SYS_STATE to SYSCONFIG_DONE.

Completion Status Codes

Table 5.216: TDH.SYS.CONFIG Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INVALID_PAMT	
TDX_INVALID_RESERVED_IN_TDMR	
TDX_INVALID_TDMR	
TDX_NON_ORDERED_RESERVED_IN_TDMR	
TDX_NON_ORDERED_TDMR	

Completion Status Code	Description
TDX_OPERAND_INVALID	
TDX_PAMT_OUTSIDE_CMRS	
TDX_PAMT_OVERLAP	
TDX_SUCCESS	TDH.SYS.CONFIG is successful.
TDX_SYS_BUSY	The operation was invoked when another TDX module operation was in progress. The operation may be retried.
TDX_SYS_CONFIG_NOT_PENDING	
TDX_SYS_SHUTDOWN	
TDX_TDMR_ALREADY_INITIALIZED	
TDX_TDMR_OUTSIDE_CMRS	

5.4.54. TDH.SYS.INFO Leaf

Provide information about the Intel TDX module and the convertible memory.

Note: TDH.SYS.INFO is provided for backward compatibility. TDH.SYS.RDALL is the recommended method to read Intel TDX module information.

Table 5.217: TDH.SYS.INFO Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address (including HKID bits) of a buffer where the output TDSYSINFO_STRUCT will be written		
RDX	The number of bytes in the above buffer		
R8	The physical address (including HKID bits) of a buffer where an array of CMR_INFO will be written		
R9	The number of CMR_INFO entries in the above buffer		

Table 5.218: TDH.SYS.INFO Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RDX	The actual number of bytes written to the above buffer In case of an error, as indicated by RAX, RDX returns 0.
R9	The number of CMR_INFO entries actually written to the above buffer In case of an error, as indicated by RAX, R9 returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.SYS.INFO provides information about the Intel TDX module and about the memory configuration.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.219: TDH.SYS.INFO Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDX system information structure	TDSYSINFO_STRUCT	RW	Shared	1024B	None	N/A	N/A
Explicit	R8	HPA	CMR table	CMR_INFO_ARRAY	RW	Shared	512B	None	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. Global and LP-scope initialization has been done:
   1.1. TDH.SYS.INIT has been executed.
   1.2. TDH.SYS.LP.INIT has been executed on the current LP.
2. The number of bytes provided for returning TDSYSINFO_STRUCT (in RDX) must be at least the size of that structure.
3. The number of entries provided for returning CMR_INFO_ARRAY (in R9) must be at least the number of CMRs supported by TDX.

If successful, the function does the following:

4. Write the TDSYSINFO_STRUCT and set RDX to the actual number of bytes written.
5. Write the CMR_INFO_ARRAY based on the CMR information in SEAMCFG and set R9 to the number of CMRs.

Completion Status Codes

Table 5.220: TDH.SYS.INFO Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDH.SYS.INFO is successful.
TDX_SYS_SHUTDOWN	
TDX_SYSINITLP_NOT_DONE	

5.4.55. TDH.SYS.INIT Leaf

Globally initialize the Intel TDX module.

Table 5.221: TDH.SYS.INIT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	Reserved		
	Bits	**Name**	**Description**
	63:0	RESERVED	Reserved: must be 0

Table 5.222: TDH.SYS.INIT Output Operands Definition

Operand	Description		
RAX	SEAMCALL instruction return code – see 5.4.1		
RCX	Extended error information part 1 If RAX returns TDX_INCORRECT_CPUID_VALUE, RCX returns the applicable CPUID information as shown below. In all other cases, RCX returns 0.		
	Bits	**Name**	**Description**
	31:0	LEAF	CPUID leaf number
	63:32	SUBLEAF	CPUID sub-leaf number: if sub-leaf is not applicable, value is -1 (0xFFFFFFFF).
RDX	Extended error information part 2 If RAX returns TDX_INCORRECT_CPUID_VALUE, RDX returns the value masks as shown below. A bit value of 1 indicates a bit position that was checked against the required value. In all other cases, RDX returns 0.		
	Bits	**Name**	**Description**
	31:0	MASK_EAX	Mask of the value returned by CPUID in EAX
	63:32	MASK_EBX	Mask of the value returned by CPUID in EBX
R8	Extended error information part 3 If RAX returns TDX_INCORRECT_CPUID_VALUE, R8 returns the value masks as shown below. A bit value of 1 indicates a bit position that was checked against the required value. In all other cases, R8 returns 0.		
	Bits	**Name**	**Description**
	31:0	MASK_ECX	Mask of the value returned by CPUID in ECX

Operand	Description		
	63:32	MASK_EDX	Mask of the value returned by CPUID in EDX
R9	Extended error information part 4 If RAX returns TDX_INCORRECT_CPUID_VALUE, R9 returns the expected values as shown below. In all other cases, R9 returns 0.		
	Bits	**Name**	**Description**
	31:0	VALUE_EAX	Value expected to be returned by CPUID in EAX
	63:32	VALUE_EBX	Value expected to be returned by CPUID in EBX
R10	Extended error information part 5 If RAX returns TDX_INCORRECT_CPUID_VALUE, R10 returns the expected values as shown below. In all other cases, R10 returns 0.		
	Bits	**Name**	**Description**
	31:0	VALUE_ECX	Value expected to be returned by CPUID in ECX
	63:32	VALUE_EDX	Value expected to be returned by CPUID in EDX
Other	Unmodified		

Special Environment Requirements

If the IA32_TSX_CTRL MSR is supported by the CPU, as enumerated by IA32_ARCH_CAPABILITIES.TSX_CTRL (bit 7), then the values of its following bits must be 0:

- RTM_DISABLE (bit 0)
- TSX_CPUID_CLEAR (bit 1)

The IA32_MISC_PACKAGE_CTRL MSR must be supported by the CPU, as enumerated by IA32_ARCH_CAPABILITIES. MISC_PACKAGE_CTRL (bit 11). IA32_MISC_PACKAGE_CTLS.ENERGY_FILTERING_ENABLE (bit 0) must be set to 1.

Leaf Function Latency

TDH.SYS.INIT execution time may be longer than most TDX module interface functions execution time. No interrupts (including NMI and SMI) are processed by the logical processor during that time.

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.SYS.INIT performs global (platform-scope) initialization of the Intel TDX module. This function is intended to be executed during OS/VMM boot and thus it has relaxed latency requirements.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.223: TDH.SYS.INIT Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	N/A	All Intel TDX module internal variables	N/A	RW	Hidden	N/A	Exclusive	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. Check that PL.SYS_STATE is SYSINIT_PENDING.
2. Do any global Intel TDX module initializations required for running this flow.
3. Check the memory operands per the table above.
4. Check the following conditions (no specific order is implied):
   - Enumerate CPU and platform information, and check Intel TDX module compatibility. If the Intel TDX module is compatible with multiple variants of CPU and platform features, sample the current LP's features enumeration – to be later checked to be the same on all LPs by TDH.SYS.LP.INIT. Examples of compatibility checks are:
     - The CPU must support any ISA that the Intel TDX module relies upon, such as SHA-NI.
     - The CPU must support the WBINVD scope for which the Intel TDX module was built.
   - Sample and check the platform configuration on the current LP – to be later checked to be the same on all LPs by TDH.SYS.LP.INIT. For example:
     - Sample SMRR and SMRR2, check they are locked and do not overlap any CMR, and store their values to be checked later on each LP.

If successful, the function does the following:

5. Complete the initialization of the Intel TDX module at platform scope.
6. Set PL.SYS_STATE to SYSINIT_DONE.

Completion Status Codes

Table 5.224: TDH.SYS.INIT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_BOOT_NT4_SET	
TDX_CPUID_LEAF_1F_FORMAT_UNRECOGNIZED	
TDX_CPUID_LEAF_1F_NOT_SUPPORTED	
TDX_CPUID_LEAF_0D_INCONSISTENT	
TDX_INCORRECT_CPUID_VALUE	Additional information is provided in RCX – R10
TDX_INCORRECT_MSR_VALUE	
TDX_INVALID_WBINVD_SCOPE	
TDX_RND_NO_ENTROPY	Random number generation (e.g., RDRAND or RDSEED) failed because the hardware random number generator did not have enough entropy. The host VMM should retry the operation.
TDX_SMRR_LOCK_NOT_SUPPORTED	
TDX_SMRR_NOT_LOCKED	
TDX_SMRR_NOT_SUPPORTED	
TDX_SMRR_OVERLAPS_CMR	
TDX_SUCCESS	TDH.SYS.INIT is successful.
TDX_SYS_BUSY	The operation was invoked when another TDX module operation was in progress. The operation may be retried.
TDX_SYS_SHUTDOWN	
TDX_SYS_INIT_NOT_PENDING	

5.4.56. TDH.SYS.KEY.CONFIG Leaf

Configure the Intel TDX global private key on the current package.

Table 5.225: TDH.SYS.KEY.CONFIG Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0

Table 5.226: TDH.SYS.KEY.CONFIG Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Latency

TDH.SYS.KEY.CONFIG execution time may be longer than most TDX module interface functions execution time. No interrupts (including NMI and SMI) are processed by the logical processor during that time.

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.SYS.KEY.CONFIG performs package-scope Intel TDX global private key configuration.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.227: TDH.SYS.KEY.CONFIG Operands Information

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	N/A	All Intel TDX module internal variables	N/A	RW	Hidden	N/A	Exclusive	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. Check that TDH.SYS.CONFIG has completed successfully (PL.SYS_STATE is SYSCONFIG_DONE).

If successful, the function does the following:

2. Do the following as an atomic operation (e.g., LOCK BTS) on PL.PKG_CONFIG_BITMAP:
   2.1. Check the package has not yet been configured.
   2.2. Mark it as configured.
3. Execute PCONFIG to configure the Intel TDX global private HKID on the package with a CPU-generated random key.

PCONFIG may fail due to an entropy error or a device busy error. In these cases, the VMM should retry TDH.SYS.KEY.CONFIG.

If successful:

4. If this was the last package on which TDH.SYS.KEY.CONFIG has executed:
    4.1. On platforms with ACT-protected memory: Enable ACT lookup.
    4.2. Set PL.STATE to SYS_READY.

Completion Status Codes

Table 5.228: TDH.SYS.KEY.CONFIG Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_KEY_CONFIGURED	
TDX_KEY_GENERATION_FAILED	Failed to generate a random key. This is typically caused by an entropy error of the CPU's random number generator, and may be impacted by RDSEED, RDRAND or PCONFIG executing on other LPs. The operation should be retried.
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. Specifically, key configuration may fail due to a concurrently running PCONFIG instruction.
TDX_SUCCESS	TDH.SYS.KEY.CONFIG is successful.
TDX_SYS_BUSY	The operation was invoked when another TDX module operation was in progress. The operation may be retried.
TDX_SYS_KEY_CONFIG_NOT_PENDING	
TDX_SYS_SHUTDOWN	

5.4.57. TDH.SYS.LP.INIT Leaf

Initialize the Intel TDX module at the current logical processor scope.

Table 5.229: TDH.SYS.LP.INIT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0

Table 5.230: TDH.SYS.LP.INIT Output Operands Definition

Operand	Description		
RAX	SEAMCALL instruction return code – see 5.4.1		
RCX	Extended error information part 1 If RAX returns TDX_INCONSISTENT_CPUID_FIELD, RCX returns the applicable CPUID information as shown below. In all other cases, RCX returns 0.		
	Bits	**Name**	**Description**
	31:0	LEAF	CPUID leaf number
	63:32	SUBLEAF	CPUID sub-leaf number: if sub-leaf is not applicable, value is -1 (0xFFFFFFFF).
RDX	Extended error information part 2 If RAX returns TDX_INCONSISTENT_CPUID_FIELD, RDX returns the value masks as shown below. A bit value of 1 indicates a bit position that was checked against the same CPUID leaf value checked during TDH.SYS.INIT. In all other cases, RDX returns 0.		
	Bits	**Name**	**Description**
	31:0	MASK_EAX	Mask of the value returned by CPUID in EAX
	63:32	MASK_EBX	Mask of the value returned by CPUID in EBX
R8	Extended error information part 3 If RAX returns TDX_INCONSISTENT_CPUID_FIELD, R8 returns the value masks as shown below. A bit value of 1 indicates a bit position that was checked against the same CPUID leaf value checked during TDH.SYS.INIT. In all other cases, R8 returns 0.		
	Bits	**Name**	**Description**
	31:0	MASK_ECX	Mask of the value returned by CPUID in ECX
	63:32	MASK_EDX	Mask of the value returned by CPUID in EDX

Operand	Description
Other	Unmodified

Special Environment Requirements

If the IA32_TSX_CTRL MSR is supported by the CPU, as enumerated by IA32_ARCH_CAPABILITIES.TSX_CTRL (bit 7), then the values of its following bits must be 0:

- RTM_DISABLE (bit 0)
- TSX_CPUID_CLEAR (bit 1)

The IA32_MISC_PACKAGE_CTRL MSR must be supported by the CPU, as enumerated by IA32_ARCH_CAPABILITIES. MISC_PACKAGE_CTRL (bit 11). IA32_MISC_PACKAGE_CTLS.ENERGY_FILTERING_ENABLE (bit 0) must be set to 1.

Leaf Function Latency

TDH.SYS.LP.INIT execution time may be longer than most TDX module interface functions execution time. No interrupts (including NMI and SMI) are processed by the logical processor during that time.

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.SYS.LP.INIT performs LP-scope initialization of the Intel TDX module. This function is intended to be executed during OS/VMM boot, and thus it has relaxed latency requirements.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.231: TDH.SYS.LP.INIT Operands Information

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	N/A	All Intel TDX module internal variables	N/A	RW	Hidden	N/A	Shared	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. TDH.SYS.INIT has completed successfully (PL.SYS_STATE is SYSINIT_DONE).
2. This is the first invocation of TDH.SYS.LP.INIT on the current LP.

If successful, the function does the following:

3. Do a global EPT flush (INVEPT type 2).
4. Initialize the Intel TDX module's LP-scope variables.
5. Check the compatibility and uniformity of features and configuration. Once per LP, core or package, depending on the scope of the checked feature or configuration:
   5.1. Check features compatibility with the Intel TDX module. In cases where the Intel TDX module supports several options, check that the features on the current LP are the same as sampled during TDH.SYS.INIT.
   5.2. Check configuration uniformity. For example, the SMRR and SMRR2 must be locked and configured in the same way as sampled during TDH.SYS.INIT.
6. Mark the current LP as initialized.

Completion Status Codes

Table 5.232: TDH.SYS.LP.INIT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INCONSISTENT_CPUID_FIELD	Additional information is provided in RCX – R8
TDX_INCONSISTENT_MSR	
TDX_INCORRECT_MSR_VALUE	
TDX_INVALID_PKG_ID	
TDX_RND_NO_ENTROPY	Random number generation (e.g., RDRAND or RDSEED) failed because the hardware random number generator did not have enough entropy. The host VMM should retry the operation.
TDX_SUCCESS	TDH.SYS.LP.INIT is successful.
TDX_SYS_BUSY	The operation was invoked when another TDX module operation was in progress. The operation may be retried.
TDX_SYS_LP_INIT_DONE	
TDX_SYS_LP_INIT_NOT_PENDING	
TDX_SYS_SHUTDOWN	

5.4.58. TDH.SYS.LP.SHUTDOWN Leaf (Deprecated)

This function is deprecated; it is provided for backward compatibility.

Table 5.233: TDH.SYS.LP.SHUTDOWN Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0

Table 5.234: TDH.SYS.LP.SHUTDOWN Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.SYS.LP.SHUTDOWN does nothing.

Completion Status Codes

Table 5.235: TDH.SYS.LP.SHUTDOWN Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_SUCCESS	TDH.SYS.LP.SHUTDOWN is successful.

5.4.59. TDH.SYS.RD Leaf

Read a TDX Module global-scope metadata field.

Table 5.236: TDH.SYS.RD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RDX	Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored. A value of -1 is a special case: it is not a valid field identifier; in this case the first readable field identifier is returned in RDX.		

Table 5.237: TDH.SYS.RD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RDX	If the input field identifier was -1, RDX returns the first readable field identifier. Else, in case of an error, RDX returns -1. On success, RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available.
R8	Contents of the field In case of no success, as indicated by RAX, R8 returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.SYS.RD reads a TDX Module global-scope metadata field.

Enumeration: Availability of TDH.SYS.RD is enumerated by TDSYSINFO_STRUCT.SYS_RD, returned by TDH.SYS.INFO (see 3.3.5). It is also enumerated by TDX_FEATURES0.ENHANCED_METADATA (bit 3), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SYS.RD returns a TDX_OPERAND_INVALID(RAX) status.

RDX returns the next host-side readable field identifier. This may be used by the host VMM to enumerate the TDX Module's capabilities and configuration. To read all the available fields, the host VMM can invoke TDH.SYS.RD in a loop, starting with field identifier -1 as an input, until RDX returns -1. A status code of TDX_METADATA_FIELD_SKIP indicates that the returned value is not applicable. Alternatively, the host VMM can use TDH.SYS.RDALL.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.238: TDH.SYS.RD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
There are no relevant memory operands.										

The function checks the following conditions:

1. Global and LP-scope initialization has been done:
   1.1. TDH.SYS.INIT has been executed.
   1.2. TDH.SYS.LP.INIT has been executed on the current LP.

If successful, the function does the following:

2. Read the requested field using the algorithm described in 5.3.2.1.
3. Return the next readable field identifier, or a value of 0 if none exists.
4. Return the field value.

Completion Status Codes

Table 5.239: TDH.SYS.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_METADATA_FIELD_ID_INCORRECT	
TDX_METADATA_FIELD_NOT_READABLE	
TDX_METADATA_FIRST_FIELD_ID_IN_CONTEXT	Indicates that the first field ID in context is returned
TDX_METADATA_FIELD_SKIP	Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any.
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDH.SYS.RD is successful.
TDX_SYS_SHUTDOWN	
TDX_SYSINITLP_NOT_DONE	

5.4.60. TDH.SYS.RDALL Leaf

Read all host-readable TDX Module global-scope metadata fields.

Table 5.240: TDH.SYS.RDALL Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RDX	The physical address (including HKID bits) of a 4KB page where a metadata list will be returned In case of error, some field value entries might not contain valid data.		
R8	Initial field identifier – see 3.10 If R8's value is -1, then TDG.SYS.RDALL will start from the first global-scope metadata field identifier. Else, LAST_ELEMENT_IN_FIELD, LAST_FIELD_IN_SEQUENCE, WRITE_MASK_VALID and CONTEXT_CODE fields are ignored. The FIELD_CODE must be the code of the first element of a metadata field.		

Table 5.241: TDH.SYS.RDALL Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
R8	Next field identifier. A value of -1 means all applicable field identifiers have been returned in the metadata list. In case of an error, as indicated by RAX, R8 returns -1.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.SYS.RDALL reads all host-readable TDX Module global-scope metadata fields into a metadata list in the provided page.

If one or more applicable fields do not fit in the provided list buffer, the function can be invoked in a loop, each invocation providing an initial field identifier returned as the next field identifier of the previous invocation, as shown in the following example:

1. NEXT_FIELD_ID = -1
2. Repeat:
   2.1. Set LIST_BUFFER to the next 4K buffer
   2.2. Invoke TDH.SYS.RDALL(RDX = LIST_BUFFER, RDX = NEXT_FIELD_ID)
   2.3. STATUS = RAX, NEXT_FIELD_ID = R8

   Until ((STATUS is a non-recoverable error) or (NEXT_FIELD_ID is -1))

The function never returns an empty list if there's no error.

Enumeration: Availability of TDH.SYS.RDALL is enumerated by TDSYSINFO_STRUCT.SYS_RD, returned by TDH.SYS.INFO (see 3.3.5). It is also enumerated by TDX_FEATURES0.ENHANCED_METADATA (bit 3), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SYS.RDALL returns a TDX_OPERAND_INVALID(RAX) status.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.242: TDH.SYS.RDALL Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RDX	HPA	Metadata list	MD_LIST	RW	Shared	4KB	None	None	None

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. Global and LP-scope initialization has been done:
   1.1. TDH.SYS.INIT has been executed.
   1.2. TDH.SYS.LP.INIT has been executed on the current LP.

If successful, the function does the following:

2. Dump the list of next host-readable metadata fields into the provided page.

Completion Status Codes

Table 5.243: TDH.SYS.RDALL Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.SYS.RDALL is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	

5.4.61. TDH.SYS.S4_END Leaf

Help prevent replay attacks, by preventing future S4 restoration using the current global S4 session's anti-replay nonce, which was generated during the S4 hibernation session.

Table 5.244: TDH.SYS.S4_END Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	SEAMCALL instruction leaf number and version, see 5.4.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the SEAMCALL interface function
		23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
		63:24	Reserved	Must be 0

Table 5.245: TDH.SYS.S4_END Output Operands Definition

Operand	Name	Description
RAX	Status	SEAMCALL instruction return code – see 5.4.1
Other		Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.SYS.S4_END generates a new anti-replay nonce. This helps prevent replay attack which may attempt to start a new S4 resumption session and resume the TDs using the anti-replay nonce generated during the S4 hibernation session.

Enumeration: Availability of TDH.SYS.S4_END is enumerated by TDX_FEATURES0.S4 (bit 13), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SYS.S4_END returns a TDX_OPERAND_INVALID(RAX) status.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.246: TDH.SYS.S4_END Operands Information

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
PL.S4_STATE	N/A	N/A	PL.S4_STATE	N/A	RW	Hidden	N/A	Exclusive	None	None

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. Check that a global S4 resumption session is in progress (PL.S4_STATE is S4_IMPORT).
2. Generate a new anti-replay NONCE by calling SEAMOPS[SEAMNONCE](0).
3. Set PL.S4_STATE to S4_IDLE.

Completion Status Codes

Table 5.247: TDH.SYS. S4_END Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_SUCCESS	TDH.SYS.S4_END is successful.
TDX_OPERAND_BUSY	
TDX_SYS_NOT_READY	TDH.SYS. S4_END was called when TDX module's lifecycle state is not SYS_READY.

5.4.62. TDH.SYS.SHUTDOWN Leaf

Initiate Intel TDX module shutdown and generate handoff data for the next Intel TDX module.

Table 5.248: TDH.SYS.SHUTDOWN Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	SEAMCALL instruction leaf number and version, see 5.4.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the SEAMCALL interface function
		23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
		63:24	Reserved	Must be 0
RCX	REQ_HV	Requested handoff version		

Table 5.249: TDH.SYS.SHUTDOWN Output Operands Definition

Operand	Name	Description
RAX	Status	SEAMCALL instruction return code – see 5.4.1
Other		Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

Enumeration: Availability of TDH.SYS.SHUTDOWN is enumerated by TDX_FEATURES0.TD_PRESERVING (bit 1), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SYS.SHUTDOWN returns a TDX_OPERAND_INVALID(RAX) status.

TDH.SYS.SHUTDOWN initiates Intel TDX module shutdown and generates handoff data for the next Intel TDX module. Following this function, no further TDX module interface functions can be called.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.250: TDH.SYS.SHUTDOWN Operands Information

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	N/A	All Intel TDX module internal variables	N/A	RW	Hidden	N/A	Exclusive	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. The requested handoff version (REQ_HV) is legal:
   1.1. PL.MIN_UPDATE_HV <= REQ_HV <= PL.MODULE_HV
   1.2. If PL.NO_DOWNGRADE == 1 then REQ_HV == PL.MODULE_HV

If successful:

2. Set TDX module's PL.STATE to SYS_SHUTDOWN to fail further TDX module interface function calls.
3. Check that all other LPs are not executing in SEAM mode.
4. Prepare handoff data in handoff pages, according to REQ_HV, from module's variables.
5. Mark the handoff data as valid (ready for consumption).

Completion Status Codes

Table 5.251: TDH.SYS.SHUTDOWN Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_INVALID	The requested handoff version is invalid.
TDX_SUCCESS	TDH.SYS.SHUTDOWN is successful.
TDX_SYS_BUSY	The operation was invoked when another TDX module operation was in progress. The operation may be retried.
TDX_SYS_NOT_READY	TDH.SYS.SHUTDOWN was called when TDX module's lifecycle state is not SYS_READY.

5.4.63. TDH.SYS.TDMR.INIT Leaf

Partially initialize a Trust Domain Memory Region (TDMR) and its associated PAMT.

Table 5.252: TDH.SYS.TDMR.INIT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical base address of a TDMR (HKID bits must be 0)		

Table 5.253: TDH.SYS.TDMR.INIT Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RDX	On successful completion, RDX returns the TDMR next-to-initialize address. This is the physical address of the last byte that has been initialized so far, rounded down to 1GB. In all other cases, RDX returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.SYS.TDMR.INIT partially initializes the metadata (PAMT) associated with a Trust Domain Memory Region (TDMR), while adhering to latency considerations. It can run concurrently on multiple LPs as long as each concurrent flow initializes a different TDMR. After each 1GB range of a TDMR has been initialized, that 1GB range becomes available for use by any Intel TDX function that creates a private TD page or a control structure page – e.g., TDH.MEM.PAGE.ADD, TDH.VP.ADDCX, etc.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.254: TDH.SYS.TDMR.INIT Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDMR	Blob	None	None	1GB	Exclusive	N/A	N/A
Implicit	N/A	HPA	PAMT region associated with TDMR	Blob	RW	Hidden	N/A	Exclusive	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. The provided TDMR start address belongs to one of the TDMRs set during TDH.SYS.INIT.
2. The TDMR has not been completely initialized yet.

If successful, the function does the following:

3. If the TDMR has been completely initialized, there is nothing to do.

Else, the function does the following:

4. On platforms which use ACT, set the ACT bit of each page containing PAMT entries to 1.
5. Initialize the next implementation defined un-initialized number of PAMT entries. The maximum number of PAMT entries to be initialized is set to help avoid latency issues.
   5.1. PAMT_4K entries associated with a physical address that is within a reserved range are marked with PT_RSVD.
   5.2. Other PAMT_4K entries are marked with PT_NDA.
   5.3. PAMT_2M and PAMT_1G entries are marked with PT_NDA.
6. If the PAMT for a 1GB block of TDMR has been fully initialized, mark that 1GB block as ready for use. This means that 4KB pages in this 1GB block may be converted to private pages – e.g., by SEAMCALL(TDH.MEM.PAGE.ADD). This can be done concurrently with initializing other TDMRs.
7. Return the next-to-initialize address rounded down to 1GB. This is done so the host VMM will not attempt to use a 1GB block that is not fully initialized.

Completion Status Codes

Table 5.255: TDH.SYS.TDMR.INIT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDH.SYS.TDMR.INIT is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TDMR_ALREADY_INITIALIZED	

5.4.64. TDH.SYS.UPDATE Leaf

Populate Intel TDX module internal variables from the handoff data prepared by the previous Intel TDX module.

Table 5.256: TDH.SYS.UPDATE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0

Table 5.257: TDH.SYS.UPDATE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

Enumeration: Availability of TDH.SYS.UPDATE is enumerated by TDX_FEATURES0.TD_PRESERVING (bit 1), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.SYS.UPDATE returns a TDX_OPERAND_INVALID(RAX) status.

TDH.SYS.UPDATE reads the handoff data prepared by the previous Intel TDX module. The operation may fail in the following cases:

- No valid handoff data
- The old module's handoff data's version is too old for the current TDX module
- The old module's handoff data's version is newer than the current TDX module's handoff data version

On such failures the host VMM is expected to do one of the following:

- Request the Persistent SEAMLDR to update to another TDX module (UPDATE scenario). If that update is successful, existing TDs are preserved.
- Keep the current TDX module and continue with the non-update sequence (TDH.SYS.CONFIG, TDH.SYS.KEY.CONFIG etc.). In this case all existing TDs are lost.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.258: TDH.SYS.UPDATE Operands Information

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	N/A	All Intel TDX module internal variables	N/A	RW	Hidden	N/A	Exclusive	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following conditions:

1. The TDX module's PL.STATE is SYSINIT_DONE.
2. All LPs have been initialized.
3. The handoff data in memory is valid and its handoff version (HV) is legal.

If successful:

4. Populate HV-specific variables within SEAM range from the handoff data.
5. Mark the handoff data as invalid (consumed).
6. Set the TDX module's PL.STATE to SYS_READY.

Completion Status Codes

Table 5.259: TDH.SYS.UPDATE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_CONNECT_INVALID_STATE	Applicable only for TDX module and CPUs that support TDX Connect
TDX_SUCCESS	TDH.SYS.UPDATE is successful.
TDX_SYS_BUSY	The operation was invoked when another TDX module operation was in progress. The operation may be retried.
TDX_SYS_STATE_INCORRECT	TDH.SYS.SHUTDOWN was called when the TDX module's lifecycle state in not SYSINIT_DONE or some LPs have not yet been initialized by TDH.SYS.LP.INIT.
TDX_SYS_INVALID_HANDOFF	The handoff data in SEAM range is invalid.

5.4.65. TDH.VP.ADDCX Leaf

Add a physical page to a TDVPS.

Table 5.260: TDH.VP.ADDCX Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of a page where the TDCX page will be added (HKID bits must be 0)		
RDX	The physical address of a TDVPR page (HKID bits must be 0)		

Table 5.261: TDH.VP.ADDCX Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.VP.ADDCX adds a physical page to a TDVPS, as a child of a given TDVPR.

Cache Lines Flushing (Future): On future platforms, if cache line flushing is required, as enumerated by TDX_FEATURES0.CLFLUSH_BEFORE_ALLOC (bit 23), then before calling TDH.VP.ADDCX, the host VMM should ensure that no cache lines associated with the added TDVPS physical page are in a Modified state, as described in the [Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.262: TDH.VP.ADDCX Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[23]
Explicit	RCX	HPA	TDCX page	Blob	RW	Opaque	4KB	Exclusive	Shared	Shared	Exclusive
Explicit	RDX	HPA	TDVPR page	Blob	RW	Opaque	4KB	Exclusive	Shared	Shared	None
Implicit	N/A	HPA	TDR page	TDR	RW	Opaque	N/A	Shared	None	None	None
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A	None

[23] ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[23]
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A	None

In addition to the explicit memory operand checks per the table above, the function checks the following conditions:

1. The TDVPR page metadata in PAMT must be correct (PT must be PT_TDVPR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD has been initialized (by TDH.MNG.INIT).
5. The TD build and measurement must not have been finalized (by TDH.MR.FINALIZE).
6. The TD VCPU has not been initialized (by TDH.VP.INIT) and is not being torn down (TDVPS.VCPU_STATE is VCPU_UNINITIALIZED).
7. The new TDCX page metadata in PAMT must be correct (PT must be PT_NDA).
8. The maximum number of TDCX pages per TDVPS (as enumerated by TDH.SYS.RD* or TDH.SYS.INFO) has not been exceeded.

If successful, the function does the following:

9. On platforms which use ACT, set the TDCX page's ACT bit to 1.
10. Zero out the TDCX page contents using direct writes (MOVDIR64B).
11. Increment the VCPU's TDCX counter and set a pointer in the parent TDVPR page to the new TDCX page.
12. Increment TDR.CHLDCNT.
13. Initialize the TDCX page metadata in PAMT.

Completion Status Codes

Table 5.263: TDH.VP.ADDCX Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.VP.ADDCX is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	
TDX_TDCX_NUM_INCORRECT	
TDX_VCPU_STATE_INCORRECT	

Intel TDX Application Binary Interface (ABI) Reference

5.4.66. TDH.VP.CREATE Leaf

Create a guest TD VCPU and its TDVPS root page (TDVPR).

Table 5.264: TDH.VP.CREATE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of a page where TDVPR will be added (HKID bits must be 0)		
RDX	The physical address of the owner TDR page (HKID bits must be 0)		

Table 5.265: TDH.VP.CREATE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.VP.CREATE begins the build of a new guest TD VCPU. It adds a TDVPR page as a child of a TDR page.

Cache Lines Flushing (Future): On future platforms, if cache line flushing is required, as enumerated by TDX_FEATURES0.CLFLUSH_BEFORE_ALLOC (bit 23), then before calling TDH.VP.CREATE, the host VMM should ensure that no cache lines associated with the added TDVPR physical page are in a Modified state, as described in the [Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.266: TDH.VP.CREATE Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions			
								Operand	Contain. 2MB	Contain. 1GB	ACT 2MB[24]
Explicit	RCX	HPA	TDVPR page	Blob	RW	Opaque	4KB	Exclusive	Shared	Shared	Exclusive
Explicit	RDX	HPA	TDR page	TDR	RW	Opaque	4KB	Shared	Shared	Shared	None
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	4KB	Shared(i)	N/A	N/A	None
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A	None

[24] ACT is enumerated by TDX_FEATURES0.ACT, readable using TDH.SYS.RD.

In addition to the explicit memory operand checks per the table above, the function checks the following conditions:

1. The TDR page metadata in PAMT must be correct (PT must be PT_TDR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must either have been initialized but not finalized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is INITIALIZED), or a migration session is in progress as state migration has begun by TDH.IMPORT.STATE.TD (OP_STATE is STATE_IMPORT).
5. The TDVPR page metadata in PAMT must be correct (PT must be PT_NDA).

If successful, the function does the following:

6. On platforms which use ACT, set the TDVPR page's ACT bit to 1.
7. Zero out the TDVPR page contents using direct write (MOVDIR64B).
8. Increment TDR.CHLDCNT.
9. Initialize the TDVPS management fields, which all reside in the TDVPR page.
10. Initialize the TDVPR page metadata in PAMT.

Completion Status Codes

Table 5.267: TDH.VP.CREATE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.VP.CREATE is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	

5.4.67. TDH.VP.ENTER Leaf

Enter TDX non-root operation.

From the host VMM's point of view, TDH.VP.ENTER is a complex operation that normally involves TD entry followed by a TD exit. Therefore, input and output operands are specified by multiple tables below.

5.4.67.1. Inputs

TDH.VP.ENTER output format depends on how the previous TDH.VP.ENTER was terminated. There are two cases:

- Initial entry or following a previous asynchronous TD exit
- Following a previous TDCALL(TDG.VP.VMCALL)

Input Format for Initial Entry or Following a Previous Asynchronous TD Exit

The following table details TDH.VP.ENTER input operands for **initial entry** or following a **previous asynchronous TD exit**.

Table 5.268: TDH.VP.ENTER Input Operands Format #1 Definition: For Initial Entry or Following a Previous Asynchronous TD Exit

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	VCPU handle and flags		
	Bit(s)	**Name**	**Description**
	11:0	RESERVED	Must be 0
	51:12	TDVPR_HPA	Bits 51:12 of the physical address of the TD VCPU's TDVPR page (HKID bits must be 0)
	52	HOST_RECOVERABILITY_HINT	Applicable only following a **previous trap-like asynchronous TD exit,** where bit 60 (HOST_RECOVERABILITY_HINT) of the previous TDH.VP.ENTER completion status (returned in RAX) was set to 1. In all other cases this bit must be 0. 0: The host VMM hints that the guest-side function may possibly be retried (e.g., the host may have corrected some conditions). 1: The host VMM hints that the error is probably not recoverable. This bit is reflected to the guest TD in bit 60 of RAX.
	53	RESUME_L1	For partitioned TDs, indicates that the L1 VMM should be resumed. Applicable after TD exits from an L2 VM. 0: TDH.VP.ENTER resumes the L2 VM it last exited from. 1: TDH.VP.ENTER resumes L1 VMM, even if the previous TD exit was from an L2 VM.
	63:54	RESERVED	Must be 0

Input Format following a Previous TDCALL(TDG.VP.VMCALL)

The following table details TDH.VP.ENTER input operands for following a **previous synchronous TD exit (TDG.VP.VMCALL)**.

Table 5.269: TDH.VP.ENTER Input Operands Format #2 Definition: Following a Previous TDCALL(TDG.VP.VMCALL)

Operand	Description		
RAX	SEAMCALL instruction leaf and version numbers – see 5.4.1		
RCX	VCPU handle and flags		
	Bit(s)	**Name**	**Description**
	11:0	RESERVED	Must be 0
	51:12	TDVPR_HPA	Bits 51:12 of the physical address of the TD VCPU's TDVPR page (HKID bits must be 0)
	52	RESERVED	Must be 0
	53	RESUME_L1	For partitioned TDs, indicates that the L1 VMM should be resumed. Applicable after TD exits from an L2 VM. 0: TDH.VP.ENTER resumes the L2 VM it last exited from. 1: TDH.VP.ENTER resumes L1 VMM, even if the previous TD exit was from an L2 VM.
	63:54	RESERVED	Must be 0
RBX, RDX, RBP, RSI, RDI, R8 – R15	If the corresponding bit of RCX at the previous TD exit (i.e., previous TDH.VP.ENTER termination) was set to 1, the register value is passed as-is to the guest TD – see the description of TDG.VP.VMCALL in 5.5.21 for details. Else, the register value is not used as an input. If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP can't be used to pass values to the guest TD. See the enumeration note below.		
XMM0 – XMM15	If the corresponding bit of RCX at the previous TD exit (i.e., previous TDH.VP.ENTER termination) was set to 1, the register value is passed as-is to the guest TD – see the description of TDG.VP.VMCALL in 5.5.21 for details. Else, the register value is not used as an input.		

5.4.67.2. Outputs

TDH.VP.ENTER output format depends on how the function was terminated. There are multiple cases:

1. Error (No TD Entry)
2. Asynchronous TD exit following a TD entry (with a VMX architectural exit reason)
3. Asynchronous TD exit following a TD entry (with a non-VMX TD exit status)
4. Asynchronous TD exit following a TD entry (with cross-TD exit details)
5. TD exit due to a TDCALL(TDG.VP.VMCALL) following a TD entry
6. TD exit due to a guest TD request

All the TD exit cases formats share some fields, as described below.

Output Format #1: Error (No TD Entry)

The following table details TDH.VP.ENTER output operands when an error occurs, and the interface function returns **without entering the TD**.

Table 5.270: TDH.VP.ENTER Output Operands Format #1 Definition: On Error (No TD Entry)

Operand	Description			
RAX	Status	SEAMCALL instruction return code		
		Bit(s)	**Name**	**Description**
		63	ERROR	Set to 1
		47:32	CLASS and DETAILS_L1	None of the values detailed in the table below
		Other		See the function completion status definition in 5.4.1.
Other GPRs	Unmodified			
Extended State	Any extended state that the TD is allowed to use (per TDCS.XFAM) may be cleared to its architectural RESET state.			

Common Output Format on TD Exits

The following table details the common format of TDH.VP.ENTER output operands when TD entry succeeds, and later a TD exit occurs. The following tables provide information for each specific case.

Table 5.271: TDH.VP.ENTER Common Output Operands Format on TD Exits Following a TD Entry

Operand	Name	Description		
RAX	Status	SEAMCALL instruction return code – see the function completion status definition in 5.4.1.		
RCX	Common Exit Information	Index of the VM from which the TD exit occurred		
		Bit(s)	**Name**	**Description**
		31:0	Format Dependent	See specific output formats below
		33:32	VM	Index of the VM that was running at the time of TD exit
		63:34	RESERVED	Reserved, set to 0
RBX, RDX, RSI, RDI, R8 – R15	Format Dependent	See specific output formats below		
RBP	None	If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP is unmodified. See the enumeration note below. Else, RBP may be modified.		
Extended State	Any extended state that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural RESET state. In case of a TDCALL(TDG.VP.VMCALL) following a TD entry, XMM may contain output operands. See below for details.			

Output Format #2: Asynchronous TD Exits Following a TD Entry (with a VMX Architectural Exit Reason)

The following table details TDH.VP.ENTER output operands when TD entry succeeds, and later an **asynchronous TD exit** occurs due to a **VMX architectural exit reason**.

Table 5.272: TDH.VP.ENTER Output Operands Format #2 Definition: On Asynchronous TD Exits Following a TD Entry (with a VMX Architectural Exit Reason)

Operand	Name	Description		
RAX	Status	SEAMCALL instruction return code		
		Bit(s)	**Name**	**Description**
		31:0	DETAILS_L2: Exit Reason	VMCS exit reason **Note:** Exit reason TDCALL (77) is a special case, indicating a synchronous TD exit initiated by TDG.VP.VMCALL; see below.
		47:32	CLASS and DETAILS_L1	May have the following values: • TDX_SUCCESS, indicating a normal TD exit • TDX_NON_RECOVERABLE_VCPU, indicating that the VCPU is disabled • TDX_NON_RECOVERABLE_TD, indicating that the TD is disabled • TDX_NON_RECOVERABLE_TD_NON_ACCESSIBLE, indicating that the TD is disabled, and its private memory can't be accessed • TDX_TD_EXIT_ON_L2_VM_EXIT and TDX_TD_EXIT_ON_L2_TO_L1, indicating a debug TD exit on L2 transitions
		Other		See the function completion status definition in 5.4.1.
RCX	Exit Information	Index of the VM from which the TD exit occurred		
		Bit(s)	**Name**	**Description**
		31:0	EXIT_QUALIFICATION	VMCS exit qualification bits 31:0 **Note:** VMCS exit qualification bits 63:32 are always 0. When exit is due to an EPT violation, bits 12:7 are cleared to 0.
		33:32	VM	Index of the VM that was running at the time of TD exit
		63:34	RESERVED	Reserved, set to 0
RDX	Extended Exit Qualification	Additional non-VMX, TDX-specific information – see 3.7.1		
R8	Guest Physical Address	When exit is due to EPT violation or EPT misconfiguration, format is similar to the VMCS guest-physical address, except that bits 11:0 are cleared to 0. In other cases, R8 is cleared to 0.		
R9	VM-Exit Interruption Information	When exit is due to a vectored event, format of bits 31:0 is similar to the VMCS VM-exit interruption information. Bits 63:32 are cleared to 0. In other cases, R9 is cleared to 0.		

Operand	Name	Description
RBX, RSI, RDI, R10 – R15	Reserved	Cleared to 0
RBP	None	If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP is unmodified. See the enumeration note below. Else, RBP is cleared to 0.
Extended State	Any extended state that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural RESET state.	

Output Format #3: Asynchronous TD Exits Following a TD Entry (with a non-VMX TD Exit Status)

The following table details TDH.VP.ENTER output operands when TD entry succeeds, and later an asynchronous TD exit occurs with a non-VMX TD exit status as described below.

Table 5.273: TDH.VP.ENTER Output Operands Format #3 Definition: On Asynchronous TD Exits Following a TD Entry (with a non-VMX TD Exit Status)

Operand	Name	Description		
RAX	Status	SEAMCALL instruction return code		
		Bit(s)	**Name**	**Description**
		47:32	CLASS and DETAILS_L1	May have the following values: • TDX_HOST_PRIORITY_BUSY_TIMEOUT • TDX_NON_RECOVERABLE_TD_CORRUPTED_MD • TDX_TD_EXIT_BEFORE_L2_ENTRY
		Other		See the function completion status definition in 5.4.1.
RCX	Exit Information	TD exit information		
		Bit(s)	**Name**	**Description**
		31:0	RESERVED	Reserved, set to 0
		33:32	VM	Index of the VM that was running at the time of TD exit
		63:34	RESERVED	Reserved, set to 0
RDX, RBX, RSI, RDI, R8 – R15	Reserved	Cleared to 0 **Note:** In the future, if this format will be used with new status codes, these GPRs may be used to return additional information.		
RBP	None	If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP is unmodified. See the enumeration note below. Else, RBP is cleared to 0.		
Extended State	Any extended state that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural RESET state.			

Output Format #4: Asynchronous TD Exits Following a TD Entry (with Cross-TD Exit Details)

The following table details TDH.VP.ENTER output operands when TD entry succeeds, and later an asynchronous TD exit occurs due to a **cross-TD operation**, i.e., the current TD operating on another TD.

Table 5.274: TDH.VP.ENTER Output Operands Format #4 Definition: On Asynchronous TD Exits Following a TD Entry (with Cross-TD Exit Details)

Operand	Name	Description		
RAX	Status	SEAMCALL instruction return code		
		Bit(s)	**Name**	**Description**
		47:32	CLASS and DETAILS_L1	May have the following values: • TDX_CROSS_TD_FAULT, indicating a fault-like asynchronous TD exit, with non-VMX cross-TD status. • TDX_CROSS_TD_TRAP, indicating a trap-like asynchronous TD exit, with non-VMX cross-TD status.
		Other		See the function completion status definition in 5.4.1.
RCX	Exit Information	TD exit information		
		Bit(s)	**Name**	**Description**
		32:0	RESERVED	Reserved, set to 0
		33:32	VM	Index of the VM that was running at the time of TD exit
		63:34	RESERVED	Reserved, set to 0
RDX	Cross-TD Status	Status code of the error which caused the TD exit, using the same format as TDCALL instruction return code		
R8	Target TD	HPA of the TDR page of the TD which was the target of the cross-TD operation		
RBX, RSI, RDI, R9 – R15	Reserved	Cleared to 0		
RBP	None	If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP is unmodified. See the enumeration note below. Else, RBP is cleared to 0.		
Extended State	Any extended state that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural RESET state.			

Output Format #5: TD Exit due to TDCALL(TDG.VP.VMCALL) Following a TD Entry

The following table details TDH.VP.ENTER output operands when TD entry succeeds, and later a **synchronous TD exit**, triggered by **TDG.VP.VMCALL**, occurs.

Table 5.275: TDH.VP.ENTER Output Operands Format #5 Definition: On TDCALL(TDG.VP.VMCALL) Following a TD Entry

Operand	Name	Description		
RAX	Status	SEAMCALL instruction return code		
		Bit(s)	**Name**	**Description**

Operand	Name	Description		
		31:0	DETAILS_L2: Exit Reason	VMCS exit reason, indicating TDCALL (77)
		47:32	CLASS and DETAILS_L1	Indicating TDX_SUCCESS
		Other		See the function completion status definition in 5.4.1.
RCX	Exit Information	TD exit information		
		Bit(s)	**Name**	**Description**
		31:0	PARAMS_MASK	Value as passed in to TDCALL(TDG.VP.VMCALL) by the guest TD: indicates which part of the guest TD GPR and XMM state is passed as-is to the VMM and back. For details, see the description of TDG.VP.VMCALL in 5.5.21.
		33:32	VM	Index of the VM that was running at the time of TD exit
		63:34	RESERVED	Reserved, set to 0
RBX, RDX, RBP, RDI, RSI, R8 – R15	GPRs	If the corresponding bit in RCX is set to 1, the register value is passed as-is from the guest TD's input to TDG.VP.VMCALL. Else, the register value cleared to 0. If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP can't be used to pass values from the guest TD and is not modified from its input value. See the enumeration note below.		
XMM0 – XMM15	XMMs	If the corresponding bit in RCX is set to 1, the register value is passed as-is from the guest TD's input to TDG.VP.VMCALL. Else, the register value cleared to 0.		
Extended State except XMM	Any extended state, except XMM, that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural RESET state.			

Output Format #6: TD Exit due to a Guest TD Request

The following table details TDH.VP.ENTER output operands when TD entry succeeds, and later on the TD requests calls a TDX module guest-side function that eventually results in a request from the host VMM.

Table 5.276: TDH.VP.ENTER Output Operands Format #6 Definition: On TD Exit due to a Guest TD Request

Operand	Name	Description		
RAX	Status	SEAMCALL instruction return code		
		Bit(s)	**Name**	**Description**
		47:32	CLASS and DETAILS_L1	May have the following values: • TDX_IOTLB_INV_REQUEST[25]
		Other		See the function completion status definition in 5.4.1.

[25] This status code is applicable if the TDX module supports TDX Connect, as enumerated by TDX_FEATURES0.TDX_CONNECT (bit 6), readable using TDH.SYS.RD*.

Operand	Name	Description		
RCX	Exit Information	TD exit information		
		Bit(s)	**Name**	**Description**
		31:0	RESERVED	Reserved, set to 0
		33:32	VM	Index of the VM that was running at the time of TD exit
		63:34	RESERVED	Reserved, set to 0
RDX, RBX, RSI, RDI, R8 – R15	Reserved	Cleared to 0 **Note:** In the future, if this format will be used with new status codes, these GPRs may be used to return additional information.		
RBP	None	If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP is unmodified. See the enumeration note below. Else, RBP is cleared to 0.		
Extended State	Any extended state that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural RESET state.			

5.4.67.3. CPU State Preservation Following a Successful TD Entry and a TD Exit

Following a successful TD entry and a TD exit, some CPU state is modified:

- Registers DR0, DR1, DR2, DR3, DR6 and DR7 are set to their architectural INIT value.
- XCR0 is set to the TD's user-mode feature bits of XFAM (bits 7:0, 9).
- MSR state preservation across TD entry and exit is detailed in a separate JSON format file **msr_preservation.json**.

5.4.67.4. Special Environment Requirements

The value read from IA32_TSC_ADJUST MSR must be the same as it was during TDH.SYS.INIT.

5.4.67.5. Guest TD State Loading or VM Entry Failure

TDH.VP.ENTER may fail loading guest TD state in the cases shown in the table below. TDH.VP.ENTER returns with information detailing the failure case. Such failures may happen due to the following reasons:

- The TD is being debugged (its ATTRIBUTES.DEBUG bit is set) and the debugger set some wrong guest state value using TDH.VP.WR. For a debuggable TD, the completion status (in RAX[63:32]) is set in such cases to TDX_SUCCESS, and the details are provided as described below. The debugger may update the VCPU state using TDH.VP.WR and invoke TDH.VP.ENTER again.
- The TD has been migrated, and some of its state is not compatible with the destination platform. The TDX module does its best effort to check guest state values during import, but there might still be cases where incompatible guest TD state gets migrated. For a non-debuggable TD, the completion status (in RAX[63:32]) is set in such cases to TDX_NON_RECOVERABLE_TD, and the details are provided as described below. The host VMM should tear down the TD.

Table 5.277: Guest State Loading Errors

Guest State Loading Error	VM Exit Reason in RAX[31:0]	Extended Exit Qualification in RDX
Error while loading guest MSR values from TDVPS	34: VM-entry failure due to MSR loading	TD_ENTRY_MSR_LOAD_FAILURE with the MSR index
Error while loading CPU extended state from TDVPS	33: VM-entry failure due to invalid guest state	TD_ENTRY_XSTATE_LOAD_FAILURE
VM entry (VMLAUNCH or VMRESUME) which loads guest state from VMCS	33: VM-entry failure due to invalid guest state	NONE

5.4.67.6. Leaf Function Latency

In some cases (e.g., suspected single/zero step attack mitigation), TDH.VP.ENTER execution time may be longer than most TDX module interface functions execution time. No interrupts (including NMI and SMI) are processed by the logical processor during that time.

5.4.67.7. Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.VP.ENTER enters TDX non-root operation. It returns immediately if TD entry failed. If TD entry succeeded, TDH.VP.ENTER returns when TD exit is initiated.

For partitioned TDs, the TD VCPU may operate in the L1 VM or one of the L2 VMs, if any. TD exit may be initiated from each of the TD's VMs. If last TD exit was from an L2 VM, TDH.VP.ENTER resumes the same L2 VM, unless the RESUME_L1 input flag is set to 1, instructing TDH.VP.ENTER to resume the L1 VM.

Enumeration: Control of RBP usage as an input/output parameter by the TD's CONFIG_FLAG.NO_RBP_MOD is enumerated by TDX_FEATURES0.NO_RBP_MOD (bit 18), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, then RBP can be used by TDG.VP.VMCALL to pass information between the guest TD and the host VMM, although highly discouraged since it contradicts normal calling conventions ABI.

VCPU Association: TDH.VP.ENTER associates the target TD VCPU with the current LP. This requires that the VCPU will not be associated with another LP. For details, see the [TDX Module Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.278: TDH.VP.ENTER Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDVPR page	TDVPS	RW	Opaque	4KB	Shared(c)[26]	Shared(c)[26]	Shared(c)[26]
Implicit	N/A	HPA	TDR page	TDR	RW	Opaque	N/A	Shared(c)[26]	N/A	N/A
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i,c)[26]	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared(t)[27]	N/A	N/A
Implicit	N/A	N/A	TDCS TLB Tracking Fields	N/A	RW	Opaque	N/A	Shared(t)[27]	N/A	N/A
Implicit	N/A	N/A	SEPT tree	N/A	R	Opaque	N/A	Exclusive(t)[27]	N/A	N/A

In addition to the explicit memory operand checks per the table above, the function checks the following conditions:

Note: For brevity, some details (e.g., zero-step mitigation) have been omitted.

1. The TDVPR page metadata in PAMT must be correct (PT must be PT_TDVPR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must have been finalized and is allowed to run (TDR.NUM_TDCX is the required number, and TDCS.OP_STATE is either RUNNING, LIVE_EXPORT or LIVE_IMPORT).

[26] The shared locking of TDVPS, TDR, TDCS, TDCS.OP_STATE is for the whole duration of running in TDX non-root mode; the locks are released on TD exit.

[27] The locking of OP_STATE, SEPT tree and the TLB tracking fields is until before entering TDX non-root mode; the locks are released before VM entry into the TD VCPU.

If successful, the function does the following:

5. Associate the VCPU with the current LP, and update TD VMCS using the algorithm described in 5.3.1.
6. If requested to resume into L1:
   6.1. Check that the current VM is an L2 VM.
   6.2. Set a sticky flag for resuming into L1.

If passed:

7. Update the TLB tracking state. This is done as a critical section allowing concurrent TDH.VP.ENTERs but no concurrent TDH.MEM.TRACK. A concurrent TDH.MEM.TRACK may cause this locking to fail; in this case, the caller is expected to retry TDH.VP.ENTER.
   7.1. Lock the TDCS epoch tracking fields in shared mode.
   7.2. Sample the TD's epoch counter (TDCS.TD_EPOCH) into the VCPU's TDVPS.VCPU_EPOCH.
   7.3. Atomically increment the TD's REFCOUNT that is associated with the sampled epoch (TDCS.REFCOUNT[TD_EPOCH % 2]).
   7.4. Release the shared mode locking of the epoch tracking fields.

If successful:

8. If TDVPS.VCPU_EPOCH was updated above, and this is not a new VCPU association:
   8.1. Execute single-context (type 1) INVEPT.
   8.2. Invalidate all soft-translated GPAs.
9. If current VM is an L2 VM:
   9.1. If the sticky flag for resuming into L1 is set:
      9.1.1. If last TD exit was synchronous (due to TDG.VP.VMCALL):
         9.1.1.1. Save the host VMM's GPR and XMM register values into TDVPS.
      9.1.2. If the sticky flag for resuming into L1 indicates a synchronous TD exit from L2:
         9.1.2.1. Translate the TDG.VP.ENTER guest state GPA.
            If failed, emulate an EPT violation TD exit.

            If passed:

         9.1.2.2. Write the TDG.VP.ENTER output to memory.
      9.1.3. Do a virtual L2→L1 exit:
         9.1.3.1. Update the GPR image in TDVPS to emulate TDG.VP.ENTER output on L2->L1 exit.
         9.1.3.2. Clear the sticky flag.
         9.1.3.3. Make L1 the current VM.
10. If entering to an L2 VM, translate soft-translated GPAs, if required.
    If failed, emulate an EPT violation TD exit.
11. If the TD VCPU to be entered is different than the last TD VCPU entered on the current LP, issue an indirect branch prediction barrier command to the CPU by writing to the IA32_PRED_CMD MSR with the IBPB bit set.
12. Set TDVPS.VCPU_STATE to VCPU_ACTIVE.
13. Restore guest TD state:
    13.1. If previous TD exit was due to a TDG.VP.VMCALL:
       13.1.1. Restore guest XMM and GPR state that is not passed as-is from the host VMM, as controlled by the value of guest TD RCX input to TDG.VP.VMCALL.
       13.1.2. Set guest RAX to 0.
    13.2. Else (TD exit was an asynchronous exit):
       13.2.1. Restore CPU extended state from TDVPS (per TDCS.XFAM).
    13.3. Restore other guest state from TDVPS.
14. Execute VMLAUNCH or VMRESUME depending on whether the entered VCPU and VM (i.e., the current VMCS) has been launched on this LP since the VCPU's last association with the LP (TDVPS.LAUNCHED[VM]).

Note: Logically, from the point of view of the host VMM, a successful TDH.VP.ENTER is terminated by the next TD exit.

5.4.67.8. Completion Status Codes

Table 5.279: TDH.VP.ENTER Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INCONSISTENT_MSR	IA32_TSC_ADJUST MSR value is different than the value sample by TDH.SYS.INIT.

Completion Status Code	Description
TDX_INCORRECT_MSR_VALUE	
TDX_L2_EXIT_HOST_ROUTED_ASYNC	
TDX_L2_EXIT_HOST_ROUTED_TDVMCALL	
TDX_NON_RECOVERABLE_TD	TDH.VP.ENTER launched or resumed TD VCPU operation (TDX non-root mode) – followed later by a TD exit. The TD state is non-recoverable – further TD entry is prohibited. Exit reason is in RAX bits 31:0.
TDX_NON_RECOVERABLE_VCPU	TDH.VP.ENTER launched or resumed TD VCPU operation (TDX non-root mode) – followed later by a TD exit. The TD VCPU state is non-recoverable – further TD entry to this VCPU is prohibited. Exit reason is in RAX bits 31:0.
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. Note the special case where the indicated operand is TLB_EPOCH. This may happen due to a conflict with TDH.MEM.TRACK or TDH.EXPORT.PAUSE. The host VMM may retry TDH.VP.ENTER. Another special case is where the indicated operand is SEPT_TREE. In some cases, TDH.VP.ENTER may acquire exclusive access on the SEPT tree for a short period of time and may fail due to a concurrent operation. The host VMM should retry TDH.VP.ENTER.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.VP.ENTER launched or resumed TD VCPU operation (TDX non-root mode) – followed later by a TD exit. Exit reason is in RAX bits 31:0.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	
TDX_TDCX_NUM_INCORRECT	
TDX_TSC_ROLLBACK	
TDX_VCPU_ASSOCIATED	
TDX_VCPU_STATE_INCORRECT	

Intel TDX Application Binary Interface (ABI) Reference

5.4.68. TDH.VP.FLUSH Leaf

Flush the address translation caches and cached TD VMCS associated with a TD VCPU on the current logical processor.

Table 5.280: TDH.VP.FLUSH Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of a TDVPR page (HKID bits must be 0)		

Table 5.281: TDH.VP.FLUSH Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.VP.FLUSH flushes the address translation caches and cached TD VMCS associated with a TD VCPU on the current LP. It then marks the VCPU as not associated with any LP.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.282: TDH.VP.FLUSH Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDVPR page	TDVPS	RW	Opaque	4KB	Exclusive	Shared	Shared
Implicit	N/A	HPA	TDR page	TDR	R	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following:

1. The TDVPR page metadata in PAMT must be correct (PT must be PT_TDVPR).
2. TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED.
3. The current VCPU must be currently associated with the current LP.

If the above checks pass, the function does the following:

4. For each L2 VM:
   4.1. Flush the TLB context and extended paging structure (EPxE) caches associated with the L2 VM using INVEPT single-context invalidation (type 1).
   4.2. Flush the cached L2 VMCS content to TDVPS using VMCLEAR.
5. Flush the TLB context and extended paging structure (EPxE) caches associated with the TD using INVEPT single-context invalidation (type 1).
6. Flush the cached TD VMCS content to TDVPS using VMCLEAR.
7. Mark the current VCPU as not associated with any LP.
8. Atomically decrement (using LOCK XADD) the associated VCPUs counter (TDCS.NUM_ASSOC_VCPUS).

Completion Status Codes

Table 5.283: TDH.VP.FLUSH Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_LIFECYCLE_STATE_INCORRECT	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.VP.FLUSH is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	
TDX_TDCX_NUM_INCORRECT	
TDX_VCPU_NOT_ASSOCIATED	

5.4.69. TDH.VP.INIT Leaf

Initialize the saved state of a TD VCPU.

Operands

Table 5.284: TDH.VP.INIT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version May be 0 or 1 (see the enumeration note below)
	63:24	Reserved	Must be 0
RCX	The physical address of a TDVPR page (HKID bits must be 0)		
RDX	Initial value of the guest TD VCPU RCX		
R8	If the version number provided in RAX[23:16] is 0, R8 is ignored. Else, R8 provides the following information:		
	Bits	**Field**	**Description**
	31:0	X2APIC_ID	VCPU's virtual x2APIC ID Must be unique across all VCPUs of the current TD.
	63:32	Reserved	Must be 0

Table 5.285: TDH.VP.INIT Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
Other	Unmodified

Leaf Function Latency

TDH.VP.INIT execution time may be longer than most TDX module interface functions execution time. No interrupts (including NMI and SMI) are processed by the logical processor during that time.

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.VP.INIT initializes the saved state of a VCPU in the TDVPR and TDPX pages.

Enumeration: TDH.VP.INIT's support of version 1 or higher is enumerated by TDX_FEATURES0.TOPOLOGY_ENUM (bit 20), readable by TDH.SYS.RD* (see 3.3.3.1), being set to 1. If not supported, calling TDH.VP.INIT with a version number higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

VCPU Association: TDH.VP.INIT associates the target TD VCPU with the current LP – for details, see the [TDX Module Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.286: TDH.VP.INIT Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDVPR page	TDVPS	RW	Opaque	4KB	Exclusive(h)	Shared	Shared
Implicit	N/A	HPA	TDR page	TDR	R	Opaque	4KB	Exclusive(h)/ Shared(h)[28]	N/A	N/A
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	4KB	Exclusive(i)/ Shared(i)[28]	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Exclusive(i)/ Shared(h)[29]	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following:

1. The TDVPR page metadata in PAMT must be correct (PT must be PT_TDVPR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must have been initialized but not finalized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is INITIALIZED).
5. The number of pages allocated to this TDVPS is correct.
6. The TD VCPU has not been initialized (by TDH.VP.INIT) and is not being torn down (TDVPS.VCPU_STATE is VCPU_UNINITIALIZED).

If successful, the function does the following:

7. Atomically increment the TD's VCPU counter (TDCS.NUM_VCPUS), and check that maximum number of VCPUs (TDCS.MAX_VCPUS) has not been exceeded.

If passed:

8. Assign VCPU_ID, a unique sequential identifier to the VCPU.
9. If TDH.VP.INIT was called with version >= 1:
   9.1. Check that the X2APIC_ID provided in R8 is different than all other VCPUs' X2APIC_ID values.

   If passed:

   9.2. Set the VCPU's virtual x2APIC_ID to the X2APIC_ID provided in R8.
10. Else (TDH.VP.INIT was called with version 0):
    10.1. Clear TDCS.TOPOLOGY_ENUM_CONFIGURED to indicate that the TD's virtual topology configuration is not valid.

If passed:

11. Initialize the VCPU state fields in the logical TDVPS structure (TDVPR and TDCX pages).
12. Associate the VCPU with the current LP and update the VMCS physical pointers and HKID execution control with the TD's HKID.
13. Set the TDVPS.LAST_TD_EXIT to ASYNC_FAULT since the first TD entry is the same as TD entry following an asynchronous fault-like TD exit.

[28] For backward compatibility, if TDH.VP.INIT is called with version == 0, then TDR is acquired in shared mode. Else, TDR is acquired in exclusive mode. TDCS is implicitly acquired with the same concurrency mode as TDR.

[29] If TDH.VP.INIT is called with version == 0, then TDCS.OP_STATE is acquired in shared mode. Else, TDCS.OP_STATE is implicitly acquired in exclusive mode, since TDR (and thus the whole TD) is acquired in exclusive mode.

Completion Status Codes

Table 5.287: TDH.VP.INIT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_MAX_VCPUS_EXCEEDED	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.VP.INIT is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCX_NUM_INCORRECT	
TDX_VCPU_ASSOCIATED	
TDX_VCPU_STATE_INCORRECT	
TDX_X2APIC_ID_NOT_UNIQUE	

5.4.70. TDH.VP.RD Leaf

Read a VCPU-scope metadata fields (control structure field) of a TD.

Table 5.288: TDH.VP.RD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Version number may be 0 or 1. See the enumeration details below.
	63:24	Reserved	Must be 0
RCX	The physical address of a TDVPR page (HKID bits must be 0)		
RDX	Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored. For TDH.VP.RD version 1 or higher, a value of -1 is a special case: it is not a valid field identifier; in this case the first readable field identifier is returned in RDX.		

Table 5.289: TDH.VP.RD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
RDX	For TDH.VP.RD version 0, RDX is unmodified. For TDH.VP.RD version 1 or higher: • If the input field identifier was -1, RDX returns the first readable field identifier. Else, in case of an error, RDX returns -1. On success, RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available.
R8	Field content In case of no success, as indicated by RAX, R8 returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.VP.RD reads a TDVPS field, given its field code. Reading is subject to the field's readability (per the TD's ATTRIBUTES.DEBUG bit).

If version 1 or higher is specified in RAX, RDX returns the next host-side readable field identifier. This may be used by the host VMM to dump the host readable VCPU metadata. To read all the available fields, the host VMM can invoke TDH.VP.RD in a loop, starting with field identifier -1 as an input, until RDX returns -1. A status code of TDX_METADATA_FIELD_SKIP indicates that the returned value is not applicable.

Enumeration: Availability of TDH.VP.RD version 1 is enumerated by TDX_FEATURES0.ENHANCED_METADATA (bit 3), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDH.VP.RD with a version number higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

VCPU Association: TDH.VP.RD associates the target TD VCPU with the current LP. This requires that the VCPU will not be associated with another LP – for details, see the [TDX Module Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.290: TDH.VP.RD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDVPR page	TDVPS	RW	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	HPA	TDR page	TDR	R	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following:

1. The TDVPR page metadata in PAMT must be correct (PT must be PT_TDVPR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must have been initialized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is not UNALLOCATED nor UNINITIALIZED).
5. The provided field code is valid.
6. The provided TDVPS field is readable per the TD's debug attribute (TDCS.ATTRIBUTES.DEBUG).

If successful, the function does the following:

7. Associate the VCPU with the current LP, and update TD VMCS using the algorithm described in 5.3.1.

If passed:

8. Read the control structure field using the algorithm described in 5.3.2.1.

Completion Status Codes

Table 5.291: TDH.VP.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_METADATA_FIELD_ID_INCORRECT	
TDX_METADATA_FIELD_NOT_READABLE	
TDX_METADATA_FIELD_SKIP	Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any.
TDX_METADATA_FIRST_FIELD_ID_IN_CONTEXT	Indicates that the first field ID in context is returned
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.

Completion Status Code	Description
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.VP.RD is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCS_NOT_ALLOCATED	
TDX_TDCX_NUM_INCORRECT	
TDX_VCPU_ASSOCIATED	
TDX_VCPU_STATE_INCORRECT	

5.4.71. TDH.VP.WR Leaf

Write a VCPU-scope metadata field (control structure field) of a TD.

Table 5.292: TDH.VP.WR Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 5.4.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the SEAMCALL interface function
	23:16	Version Number	Selects the SEAMCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	The physical address of a TDVPR page (HKID bits must be 0)		
RDX	Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored.		
R8	64b value to write to the field		
R9	A 64b write mask to indicate which bits of the value in R8 are to be written to the field		

Table 5.293: TDH.VP.WR Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 5.4.1
R8	Previous content of the field In case of an error, as indicated by RAX, R8 returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.VP.WR writes a TDVPS field, given its field code. The value (R8) is written as specified by the write mask (R9). Writing is subject to the field's internal write mask (per the TD's ATTRIBUTES.DEBUG bit). Writing of specific fields is also subject to additional rules as detailed in 4.2.

Table 5.294: Metadata Field Write Rules

Write Mask Bit in R9	Internal Write Mask Bit	Value Bit in R8
0	N/A	Silently ignored
1	0	Must be the same as the current field's bit
1	1	Written to the current field's bit

TDH.VP.WR returns the previous content of the field masked by the field's readability (per the TD's ATTRIBUTES.DEBUG bit).

VCPU Association: TDH.VP.WR associates the target TD VCPU with the current LP. This requires that the VCPU will not be associated with another LP – for details, see the [TDX Module Base Spec].

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.295: TDH.VP.WR Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDVPR page	TDVPS	RW	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	HPA	TDR page	TDR	R	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared	N/A	N/A

In addition to the memory operand checks per the table above, the function checks the following:

1. The TDVPR page metadata in PAMT must be correct (PT must be PT_TDVPR).
2. The TD is not in a FATAL state (TDR.FATAL is FALSE).
3. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
4. The TD must have been initialized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is not UNALLOCATED nor UNINITIALIZED).
5. The provided field code is valid.
6. The provided TDVPS field is writable per the TD's debug attribute (TDCS.ATTRIBUTES.DEBUG).

If successful, the function does the following:

7. Associate the VCPU with the current LP, and update TD VMCS using the algorithm described in 5.3.1.

If passed:

8. Write the control structure field and return its old value, using the algorithm described in 5.3.2.2.
   - 8.1. Writes of some fields are subject to rules, as detailed per field in 4.2 – e.g., the value of fields that contain Shared physical address, such as the Shared EPT Pointer, must have a Shared HKID value and must comply with some alignment rules.
   - 8.2. In most cases, writes of guest state fields are subject to the same rules as if the write is done by the guest itself – e.g., writing to guest CR4 is subject to the rules described in the [TDX Module Base Spec]. If the write operation is invalid, TDH.VP.WR fails and returns a proper error code.
   - 8.3. In debug mode (ATTRIBUTES.DEBUG == 1), there are some TDVPS fields where the TDH.VP.WR does not check whether the written values are architecturally valid. It is the responsibility of the host VMM, and failing to do so will later cause a VM entry failure leading to a fatal shutdown of the Intel TDX module. The security of any guest TD is not impacted.
   - 8.4. In other cases, in debug mode (ATTRIBUTES.DEBUG == 1), TDH.VP.WR allows setting of TDVPS fields to values that may impact the correct operation of the TD under debug. It is the responsibility of the host VMM to take this into consideration.
     - TDH.VP.WR is allowed to enable BTM by setting guest IA32_DEBUGCTL[7:6] to 0x1.
     - TDH.VP.WR is allowed to modify the state of IA32_DEBUCTL[13] (ENABLE_UNCORE_PMI).
     - TDH.VP.WR is allowed to enable VM exits on exceptions other than MCE by setting the TD VMCS exception bitmap execution control. The Intel TDX module does not take this into account when handling VM exits that occur during event delivery.

Completion Status Codes

Table 5.296: TDH.VP.WR Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_METADATA_FIELD_ID_INCORRECT	

Completion Status Code	Description
TDX_METADATA_FIELD_NOT_READABLE	
TDX_METADATA_FIELD_NOT_WRITABLE	
TDX_METADATA_FIELD_VALUE_NOT_VALID	
TDX_METADATA_WR_MASK_NOT_VALID	
TDX_OP_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.VP.WR is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_VMCS_FIELD_NOT_INITIALIZED	
TDX_TDCS_NOT_ALLOCATED	
TDX_TDCX_NUM_INCORRECT	
TDX_TDVPS_FIELD_NOT_WRITABLE	
TDX_VCPU_ASSOCIATED	
TDX_VCPU_STATE_INCORRECT	
TDX_TD_VMCS_FIELD_NOT_INITIALIZED	

5.5. Guest-Side (TDCALL) Interface Functions

The TDCALL instruction causes a VM exit to the Intel TDX module. It is used to call guest-side Intel TDX functions, either local or a TD exit to the host VMM, as selected by RAX.

5.5.1. TDCALL Instruction (Common)

This section describes the common functionality of TDCALL. Leaf functions are described in the following sections. As used by the Intel TDX module, TDCALL is allowed only in 64b mode.

Table 5.297: TDCALL Input Operands Definition

Operand	Description		
RAX	Leaf and version numbers, as defined in the [TDX Module Base Spec]. See Table 5.299 below for TDCALL leaf numbers.		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version
	63:24	Reserved	Must be 0
Other	See individual TDCALL leaf functions.		

Table 5.298: TDCALL Output Operands Definition

Operand	Description
RAX	Instruction return code, indicating the outcome of execution of the instruction – see the [TDX Module Base Spec] for details.
Other	See individual TDCALL leaf functions.

Table 5.299: TDCALL Instruction Leaf Numbers Definition

Leaf #	Interface Function Name	Description
0	TDG.VP.VMCALL	Call a host VM service
1	TDG.VP.INFO	Get TD execution environment information
2	TDG.MR.RTMR.EXTEND	Extend a TD run-time measurement register
3	TDG.VP.VEINFO.GET	Get Virtualization Exception Information for the recent #VE exception
4	TDG.MR.REPORT	Creates a cryptographic report of the TD
5	TDG.VP.CPUIDVE.SET	Control delivery of #VE on CPUID instruction execution
6	TDG.MEM.PAGE.ACCEPT	Accept a pending private page into the TD
7	TDG.VM.RD	Read a TD-scope metadata field
8	TDG.VM.WR	Write a TD-scope metadata field
9	TDG.VP.RD	Read a VCPU-scope metadata field
10	TDG.VP.WR	Write a VCPU-scope metadata field
11	TDG.SYS.RD	Read a TDX Module global-scope metadata field
12	TDG.SYS.RDALL	Read all gust-readable TDX Module global-scope metadata fields
18	TDG.SERVTD.RD	Read a target TD metadata field
20	TDG.SERVTD.WR	Write a target TD metadata field
22	TDG.MR.VERIFYREPORT	Verify a cryptographic report of a TD, generated on the current platform
23	TDG.MEM.PAGE.ATTR.RD	Read the GPA mapping and attributes of a TD private page
24	TDG.MEM.PAGE.ATTR.WR	Write the attributes of a private page
25	TDG.VP.ENTER	Enter L2 VCPU operation

Leaf #	Interface Function Name	Description
26	TDG.VP.INVEPT	Invalidate cached EPT translations for selected L2 VMs
27	TDG.VP.INVGLA	Invalidate cached translations for selected pages in an L2 VM

Instruction Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

This section describes how TDCALL leaf functions are implemented by the Intel TDX module.

The TDCALL instruction itself is specified in the [TDX Arch Extensions Spec]. There are multiple cases where TDCALL may fail. Failures may result in an exception (#UD, #GP(0)). Failure cases include, among other, the following:

- CPU mode is incorrect
- Privilege level is not 0

TDCALL results in a VM exit to the TDX module. On VM exit, the Intel TDX module performs the following checks:

1. If the CPU mode is not 64b ((IA32_EFER.LMA == 1) && (CS.L == 1)), the Intel TDX module injects a #GP(0) fault into the guest TD.
2. If the leaf number in RAX is not supported by the Intel TDX module, it returns a TDX_OPERAND_INVALID(0) status code in RAX.

If all checks pass, the Intel TDX module calls the leaf function according to the leaf number in RAX – see the following sections for individual leaf function details.

Completion Status Codes

Table 5.300: TDCALL Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_SUCCESS	TDCALL is successful.
TDX_OPERAND_INVALID	Invalid leaf number
Other	See individual leaf functions

5.5.2. TDG.MEM.PAGE.ACCEPT Leaf

Accept a pending private page and initialize it to all-0 using the TD ephemeral private key.

Table 5.301: TDG.MEM.PAGE.ACCEPT Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the Secure EPT leaf entry that maps the private page to be accepted: either 0 (4KB) or 1 (2MB) – see 3.6.1.
	11:3	Reserved	Reserved: must be 0
	51:12	GPA	Bits 51:12 of the guest physical address of the private page to be accepted
	63:52	Reserved	Reserved: must be 0

Table 5.302: TDG.MEM.PAGE.ACCEPT Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 5.5.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.MEM.PAGE.ACCEPT accepts a pending private page, previously added by TDH.MEM.PAGE.AUG, into the TD. It initializes the page to 0. If page attributes have been set by the guest TD, using TDG.MEM.PAGE.ATTR.WR, while the page was pending, they become effective when the page is accepted.

SEPT Mapping Size Considerations

In most cases, the guest TD is unaware of how TD private pages are mapped by the host VMM in SEPT. However, TDG.MEM.PAGE.ACCEPT operation specifies a page mapping size and may fail if the specified size is different than the actual mapping size.

- If the page is mapped at a lower level than requested, the function returns TDX_PAGE_SIZE_MISMATCH. The guest may re-invoke TDG.MEM.PAGE.ACCEPT specifying a 4KB page size.
- If the page is mapped at a higher level than requested, this results in an EPT violation TD exit, with extended exit qualification indicating the error SEPT entry level and state, and the guest-requested mapping level. The host VMM is expected to demote the page, then re-enter the guest TD so TDG.MEM.PAGE.ACCEPT is re-invoked.

Other Conditions that Prevent Page Acceptance

- If the page has already been accepted, the function returns TDX_PAGE_ALREADY_ACCEPTED.
- If the page is not PENDING nor PENDING_EXPORTED_DIRTY, this result in an EPT violation TD exit, with extended exit qualification indicating the error SEPT entry level and state, and the guest-requested accept level.

Interruptibility

If, during its execution, TDG.MEM.PAGE.ACCEPT detects that an external interrupt is pending, it may resume the guest TD with the CPU state unmodified. The progress so far is recorded in the page's Secure EPT entry. This allows the external interrupt to be recognized, causing a TD exit or a posted interrupt delivery. Typically, TDG.MEM.PAGE.ACCEPT will be re-invoked and continue its work.

Guest TD software is not directly involved. Guest TD should not precede the TDCALL with an STI instruction or a MOV to SS instruction. Posted interrupts may be delivered when the TDCALL flow is interrupted.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.303 TDG.MEM.PAGE.ACCEPT Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Explicit	RCX	GPA	TD private page	Blob	RW	Private	$2^{12+9*Level}$ Bytes	None
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDVPS structure	TDVPS	RW	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	Secure EPT tree	N/A	RW	Private	N/A	None
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive(h)

TDG.MEM.PAGE.ACCEPT checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

In addition to the memory operand checks per the table above, the function does the following (no specific order is implied):

1. Walk the Secure EPT based on the GPA operand and requested level. The walk is successful if arrived at a leaf entry whose state is PENDING. In case of error, return a status code or TD exit as described in the [TDX Module Base Spec].

If successful, do the following:

2. Loop until the whole page has been initialized, or until interrupted:
   2.1. Initialize the next 4KB chunk to 0 using the TD's ephemeral private HKID and direct writes (MOVDIR64B).
   2.2. If not done and there is a pending interrupt, abort TDG.MEM.PAGE.ACCEPT and resume the guest TD without updating RIP and any GPR.

If done initializing the page, do the following:

3. Set the SEPT entry to MAPPED.
4. For each L2 VM where the page is mapped:
   4.1. Walk the L2 Secure EPT based on the GPA operand and find the L2 SEPT entry for the page to be accepted.
   4.2. Restore the L2 SEPT entry attributes.
   4.3. Set the L2 SEPT entry state to L2_MAPPED.

Completion Status Codes

Table 5.304: TDG.MEM.PAGE.ACCEPT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_INVALID	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation. Specifically, it may indicate that a concurrent TDG.MEM.PAGE.ACCEPT is using the same Secure EPT entry
TDX_PAGE_ALREADY_ACCEPTED	
TDX_PAGE_SIZE_MISMATCH	Requested page size is 2MB, but the page GPA is not mapped at 2MB size
TDX_SUCCESS	TDG.MEM.PAGE.ACCEPT is successful.

5.5.3. TDG.MEM.PAGE.ATTR.RD Leaf

Read the GPA mapping and attributes of a TD private page or a private MMIO page[30].

Table 5.305: TDG.MEM.PAGE.ATTR.RD Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	TDCALL instruction leaf number and version, see 5.5.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the TDCALL interface function
		23:16	Version Number	Selects the TDCALL interface function version Must be 0
		63:24	Reserved	Must be 0
RCX	GPA	Guest physical address		

Table 5.306: TDG.MEM.PAGE.ATTR.RD Output Operands Definition

Operand	Name	Description		
RAX	STATUS	TDCALL instruction return code – see 5.5.1		
RCX	GPA_MAPPING	Actual GPA mapping of the page:		
		Bits	**Name**	**Description**
		2:0	LEVEL	Level of the Secure EPT leaf entry that maps the private page: either 0 (4KB), 1 (2MB) or 2 (1GB) – see 3.6.1.
		11:3	RESERVED	Reserved: set to 0
		51:12	GPA	Bits 51:12 of the guest physical start address of the private page Depending on the level, the following least significant bits are always 0: Level 0 (EPTE): None Level 1 (EPDE): Bits 20:12 Level 2 (EPDPTE): Bits 29:12
		61:52	RESERVED	Reserved: set to 0
		62	PENDING	Flags that the page is PENDING This is applicable to all PENDING states; if the TDX module supports TDX Connect, it is also applicable to MMIO_PENDING pages.
		63	RESERVED	Reserved: set to 0
RDX	GPA_ATTR	Guest-visible page attributes. See the GPA_ATTR definition in 3.6.3.		
Other		Unmodified		

[30] Applicable only if the TDX module supports TDX Connect.

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.PAGE.ATTR.RD reads the GPA mapping and attributes of a TD private page. Given a GPA (which can be anywhere within a page) it returns the actual mapping level – either 0 (4KB), 1 (2MB) or 2 (1GB) – and the guest-readable attributes. TDH.MEM.PAGE.ATTR.RD can read the attributes of a PENDING page.

GPA mapping level is exposed to the guest TD since page acceptance (TDH.MEM.PAGE.ACCEPT) and page attributes modifications and L2 page aliases management (TDH.MEM.PAGE.ATTR.WR) are done at mapping granularity.

Enumeration: Availability of TDG.MEM.PAGE.ATTR.RD is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.MEM.PAGE.ATTR.RD returns a TDX_OPERAND_INVALID(RAX) status.

Support of TDX Connect is enumerated by TDX_FEATURES0.TDX_CONNECT (bit 6) and TDX_FEATURES0.TDX_CONNECT_PARTITIONING (bit 32).

EPT Violation: If the requested GPA is not guest-readable and not pending acceptance, TDH.MEM.PAGE.ATTR.RD causes an EPT violation TD exit.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.307 TDG.MEM.PAGE.ATTR.RD Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Explicit	RCX	GPA	TD private page	Blob	RW	Private	None	None
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	Secure EPT tree	N/A	R	Private	N/A	None
Implicit	N/A	N/A	L2 Secure EPT trees	N/A	RW	Private	N/A	None
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	R	Private	N/A	Exclusive(h)
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)

TDG.MEM.PAGE.ATTR.RD checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

In addition to the memory operand checks per the table above, the function does the following (no specific order is implied):

1. Check that the requested GPA is valid.
2. Walk the L1 Secure EPT based on the GPA operand.
   2.1. The walk is successful if arrived at a leaf entry whose state is either guest accessible (MAPPED, EXPORTED_DIRTY, *BLOCKEDW*) or pending but not blocked (PENDING, or PENDING_EXPORTED_*).
   2.2. Else, do an EPT violation TD exit.

If successful, do the following:

3. For each L2 page alias to the L1 SEPT entry:
   3.1. Walk that L2 VM's SEPT and locate the page alias L2 SEPT leaf entry.
   3.2. Read the L2 SEPT entry and assemble the returned attributes. If the page is pending or blocked for writing, the L2 SEPT entry's original access permission bits are read from their saved locations in the L2 SEPT entry.
4. Return the page mapping and attributes.

Completion Status Codes

Table 5.308: TDG.MEM.PAGE.ATTR.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.MEM.PAGE.ATTR.RD is successful.

5.5.4. TDG.MEM.PAGE.ATTR.WR Leaf

Write the attributes of a private page or a private MMIO page[31]. Create or remove L2 page aliases as required.

Table 5.309: TDG.MEM.PAGE.ATTR.WR Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	TDCALL instruction leaf number and version, see 5.5.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the TDCALL interface function
		23:16	Version Number	Selects the TDCALL interface function version Must be 0
		63:24	Reserved	Must be 0
RCX	GPA_MAPPING	GPA mapping information:		
		Bits	**Name**	**Description**
		2:0	LEVEL	Level of the Secure EPT leaf entry that maps the private page: either 0 (4KB), 1 (2MB) or 2 (1GB) – see 3.6.1.
		11:3	RESERVED	Reserved: must be 0
		51:12	GPA	Bits 51:12 of the guest physical address of the private page Depending on the level, the following least significant bits must be 0: Level 0 (EPTE): None Level 1 (EPDE): Bits 20:12 Level 2 (EPDPTE): Bits 29:12
		63:52	RESERVED	Reserved: must be 0
RDX	GPA_ATTR	Guest-visible page attributes. See the GPA_ATTR definition in 3.6.3. To avoid writing the attributes of a certain VM, all 16 attribute bits (GPA_ATTR_SINGLE_VM) for that VM should be set to 0. Attribute bits for non-existent VMs must be 0.		
R8	ATTR_FLAGS	Attributes masks and invalidate EPT flags		
		Bits	**Field**	**Description**
		15:0	RESERVED	Must be 0
		30:16	ATTR_MASK1	A bit value of 1 indicates that the applicable attributes bit is to be written. Otherwise, the attributes bit is unmodified. Must be 0 if the TD has no VM #1.
		31	INVEPT1	Invalidate EPT for L2 VM #1 Must be 0 if the TD has no VM #1.
		46:32	ATTR_MASK2	A bit value of 1 indicates that the applicable attributes bit is to be written. Otherwise, the attributes bit is unmodified.

[31] Applicable only if the TDX module supports TDX Connect.

Operand	Name	Description		
				Must be 0 if the TD has no VM #2.
		47	INVEPT2	Invalidate EPT for L2 VM #2 Must be 0 if the TD has no VM #2.
		62:48	ATTR_MASK3	A bit value of 1 indicates that the applicable attributes bit is to be written. Otherwise, the attributes bit is unmodified. Must be 0 if the TD has no VM #3.
		63	INVEPT3	Invalidate EPT for L2 VM #3 Must be 0 if the TD has no VM #3.

Table 5.310: TDG.MEM.PAGE.ATTR.WR Output Operands Definition

Operand	Name	Description		
RAX	STATUS	TDCALL instruction return code – see 5.5.1		
RCX	GPA_MAPPING	Actual GPA mapping of the page:		
		Bits	**Name**	**Description**
		2:0	LEVEL	Level of the Secure EPT leaf entry that maps the private page: either 0 (4KB), 1 (2MB) or 2 (1GB) – see 3.6.1.
		11:3	RESERVED	Reserved: set to 0
		51:12	GPA	Bits 51:12 of the guest physical start address of the private page Depending on the level, the following least significant bits are always 0: Level 0 (EPTE): None Level 1 (EPDE): Bits 20:12 Level 2 (EPDPTE): Bits 29:12
		61:52	RESERVED	Reserved: set to 0
		62	PENDING	Flags that the page is PENDING This is applicable to all PENDING states; if the TDX module supports TDX Connect, it is also applicable to MMIO_PENDING pages.
		63	RESERVED	Reserved: set to 0
RDX	GPA_ATTR	On success, if the attribute bits (GPA_ATTR_SINGLE_VM) for a specific VM were 0 on input, they remain unmodified. For other VMs, RDX returns the updated guest-visible page attributes. In case of an error, RDX returns the current value of page attributes when possible. If the current attributes for a certain VM have not been read, that VM's attributes VALID bit returns 0. See the GPA_ATTR definition in 3.6.3.		
Other		Unmodified		

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDH.MEM.PAGE.ATTR.WR writes the specified set of attributes of a TD private page, including L2 page alias attributes. Only the bits selected by the attributes mask are updated. The private page can be either writable by the TD (MAPPED or EXPORTED_DIRTY) or pending acceptance (PENDING or PENDING_EXPORTED_DIRTY). If the page is pending acceptance, the written attributes will become effective when the page is later accepted by the guest TD, using TDG.MEM.PAGE.ACCEPT.

TDH.MEM.PAGE.ATRR.WR ignores any VM's GPA attributes set whose VALID bit is 0.

TDH.MEM.PAGE.ATTR.WR creates or removes L2 page aliases as required:

- If any of the requested L2 attributes VALID bit is set, and the R, W, Xs, Xu and PWA bits combination has a legal, non-0 value, then if the L2 page alias does not exist, it is created. The rules for checking the legal combination of attributes bits are described in 3.6.3.1.
- If any of the requested L2 attributes VALID bit is set, and the R, W, Xs, Xu and PWA bits are all 0, then if the L2 page alias exists, it is removed.

Note: For the above operations, the Xu and PWA bits are always considered, regardless of the L2 VMCS setting of the "mode-based execute control for EPT" (MBEC) and "EPT paging-write control" VM-execution controls.

Enumeration: Availability of TDG.MEM.PAGE.ATTR.WR is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.MEM.PAGE.ATTR.WR returns a TDX_OPERAND_INVALID(RAX) status.

Support of TDX Connect is enumerated by TDX_FEATURES0.TDX_CONNECT (bit 6) and TDX_FEATURES0.TDX_CONNECT_PARTITIONING (bit 32).

SEPT Mapping Size Considerations

In most cases, the guest TD is unaware of how TD private pages are mapped by the host VMM in SEPT. However, TDG.MEM.PAGE.ATTR.WR operation specifies a page mapping size and may fail if the specified size is different than the actual mapping size.

- If the page is mapped at a lower level than requested, the function returns TDX_PAGE_SIZE_MISMATCH. The guest may re-invoke TDG.MEM.PAGE.ATTR.WR specifying the actual mapping size as returned in RCX.
- If the page is mapped at a higher level than requested, this results in an EPT violation TD exit, with extended exit qualification indicating the error SEPT entry level and state, and the guest-requested mapping level. The host VMM is expected to demote the page, then re-enter the guest TD so TDG.MEM.PAGE.ATTR.WR is re-invoked.

Other Conditions that Prevent Page Attributes Modifications

- If the page is not guest-writable and is not pending, this results in an EPT violation TD exit, indicating a failed write operation.
- If an L2 SEPT walk fails, meaning there's a missing non-leaf L2 SEPT page, the operation depends on the setting of the host writable TDCS field VM_CTLS, which is an array of 4 bitmaps, one per VM (only L2 VMs are applicable). Bit 0 controls the operation on L2 SEPT walk fails in TDCALL flows:
  - The default value of 0 means that a TDX_L2_SEPT_WALK_FAILED status is returned to the L1 VMM.
  - If the value is 1, the TDX module does an EPT violation TD exit, indicating a failed write operation exit, with extended exit qualification indicating the error L2 SEPT level and VM index. The host VMM may then add the missing L2 SEPT page using TDH.MEM.SEPT.ADD.

In any of the above cases, the page attributes are not modified.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.311 TDG.MEM.PAGE.ATTR.WR Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Explicit	RCX	GPA	TD private page	Blob	RW	Private	$2^{12+9*Level}$ Bytes	None

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	L1 Secure EPT tree	N/A	R	Private	N/A	None
Implicit	N/A	N/A	L2 Secure EPT trees	N/A	RW	Private	N/A	None
Implicit	N/A	GPA	L1 Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive(h)
Implicit	N/A	GPA	L2 Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive(i)

TDG.MEM.PAGE.ATTR.WR checks the memory operands per the table above when applicable during its flow. The text below does not explicitly mention those checks, except when necessary.

In addition to the memory operand checks per the table above, the function does the following (no specific order is implied):

1. Check that the requested page attributes and attributes mask operands are valid:
   1.1. Non-0 attributes and mask are only allowed for existing L2 VMs.
2. Check that the requested GPA and level are valid.

If successful, do the following:

3. Walk the L1 Secure EPT based on the GPA operand and requested level.
   3.1. If failed, do an EPT violation TD exit, indicating a failed write operation. Extended exit information provides the host VMM with details.
   3.2. If arrived at a non-leaf entry, return a TDX_PAGE_SIZE_MISMATCH status.
      3.2.1. The guest TD may request the host VMM to demote the page mapping.

If passed:

4. Check SEPT entry state. TDG.MEM.PAGE.ATTR.WR is allowed for guest-writable or non-blocked pending leaf pages.
   4.1. If failed, do an EPT violation TD exit, indicating a failed write operation. Extended exit information provides the host VMM with details.

If passed, check and prepare the update L2 attributes:

5. For each L2 VM:
   5.1. Calculate the combined attributes value for this L2 VM based on the provided attributes and mask.
   5.2. If the combined attributes are valid:
      5.2.1. If an L2 page alias exists, find the existing page alias L2 SEPT leaf entry:
         5.2.1.1. Walk that L2 VM's SEPT and locate the page alias L2 SEPT leaf entry.
         5.2.1.2. Get the current L2 attributes and calculate the effective new attributes to be updated.
         5.2.1.3. Check that the new attributes are legal.
      5.2.2. Else (L2 page alias does not exist), if the combined attributes indicate that an L2 alias should be created:
         5.2.2.1. Walk that L2 VM's SEPT and locate the page alias L2 SEPT leaf entry.
         5.2.2.2. If failed, then depending on the setting of TDCS.VM_CTLS either do an EPT violation TD exit or return a status code.

If all checks passed, commit the updates:

6. For each L2 VM:
   6.1. If an L2 alias exists:
      6.1.1. If the combined attributes are valid, update the existing L2 alias leaf SEPT entry:
         6.1.1.1. If the combined attributes indicate that an L2 mapping is present, update the L2 SEPT entry.
         6.1.1.2. Else, mark the L2 SEPT entry as free.
   6.2. Else (L2 alias dons exist),
      6.2.1. If an existing L2 was found earlier, update it.
   6.3. If requested by the INVEPT flag, flush the TLB context and extended paging structure (EPxE) caches associated with the L2 VM, using INVEPT single-context invalidation (type 1).

Completion Status Codes

Table 5.312: TDG.MEM.PAGE.ATTR.WR Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_INVALID	
TDX_OPERAND_BUSY	
TDX_PAGE_ATTR_INVALID	The combination of page attributes to be set, after considering the requested attributes, the requested attributes mask and the current page attributes, is invalid.
TDX_PAGE_SIZE_MISMATCH	Requested page size does not match its GPA mapping size
TDX_SUCCESS	TDG.MEM.PAGE.ATTR.WR is successful.

5.5.5. TDG.MR.REPORT Leaf

TDG.MR.REPORT creates a TDREPORT_STRUCT structure that contains the measurements/configuration information of the guest TD that called the function, measurements/configuration information of the Intel TDX module and a REPORTMACSTRUCT.

Table 5.313: TDG.MR.REPORT Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	Guest physical address of newly created report structure. • For version 0, the buffer must be aligned on 1024B.		
RDX	64B-aligned guest physical address of additional data to be signed		
R8	**Bits**	**Name**	**Description**
	7:0	Report sub type	Must be 0
	63:8	Reserved	Reserved: must be 0

Table 5.314: TDG.MR.REPORT Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 5.5.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

This function creates a TDREPORT_STRUCT structure that contains the measurements/configuration information of the guest TD that called the function, measurements/configuration information of the Intel TDX module and a REPORTMACSTRUCT. The REPORTMACSTRUCT is integrity-protected with a MAC, and it contains the hash of the measurements and configuration as well as additional REPORTDATA provided by the TD software.

The created TDREPORT_STRUCT version (REPORTTYPE.VERSION) is the lowest version that contains all the TD's reported information:

- If any service TD has been bound or pre-bound (i.e., SERVTD_HASH is not 0), then the version is 1.
- Else, the version is 0.

Additional REPORTDATA, a 64-byte value, is provided by the guest TD to be included in the TDG.MR.REPORT.

Note: Although not enforced by TDG.MR.REPORT, the guest TD should normally place REPORTDATA in private memory to help ensure secure report generation.

Interruptibility If, during its execution, TDG.MR.REPORT detects that an external interrupt is pending, it may resume the guest TD with the CPU state unmodified. The progress so far is recorded internally. This allows the

external interrupt to be recognized, causing a TD exit or a posted interrupt delivery. Typically, TDG.MR.REPORT will be re-invoked and continue its work.

Guest TD software is not directly involved. Guest TD should not precede the TDCALL with an STI instruction or a MOV to SS instruction. Posted interrupts may be delivered when the TDCALL flow is interrupted.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.315: TDG.MR.REPORT Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Explicit	RCX	GPA	Output report	TDREPORT_STRUCT	RW	Private/ Shared	1024B	None
Explicit	RDX	GPA	Input report data	REPORTDATA	R	Private/ Shared	64B	None
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS.RTMR	SHA384_HASH	N/A	Opaque	N/A	Shared
Implicit	N/A	N/A	TDVPS structure	TDVPS	None	Opaque	N/A	Shared(i)

In addition to the memory operand checks per the table above, the function checks the following conditions (no specific order is implied):

1. R8 must specify report sub type 0.

If passed:

2. Determine REPORTTYPE.VERSION:
   2.1. If there is any bound or pre-bound service TDs, then REPORTTYPE.VERSION is 1.
   2.2. Else REPORTTYPE.VERSION is 0.

If passed:

3. Assemble a REPORTTYPE structure.
4. Assemble the output report's TDINFO_STRUCT base fields from the TDCS reported fields (ATTRIBUTES, XFAM, MRTD, MRCONFIGID, MROWNER, MROWNERCONFIG and RTMRs).
5. If REPORTTYPE.VERSION is 0, add the SERV_TD hash field as 0, and the RESREVED field.
6. If REPORTTYPE.VERSION is 1, add the SERV_TD hash field from TDCS, and the RESREVED field.
7. Calculate a SHA384 hash over TDINFO (size depends on REPORTTYPE.VERSION).
8. If the TDX module supports TD preserving updates:
   8.1. Execute SEAMOPS(SEAMDB_REPORT) to complete the output report, based on the input report data, the TDINFO hash calculated above, the report type structure and the SEAMDB entry's index/nonce pair of the TDR.
   8.2. If SEAMDB_REPORT returns an error (unrecognized index/nonce pair), then mark the TD state as FATAL and do a TD exit with a TDX_NON_RECOVERABLE_TD_CORRUPTED_MD status code.
9. Else, execute SEAMOPS(SEAM_REPORT) to complete the output report, based on the input report data, the TDINFO hash calculated above and the report type structure.

If successful:

10. Write the output report to memory.

Completion Status Codes

Table 5.316: TDG.MR.REPORT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.MR.REPORT is successful.

5.5.6. TDG.MR.RTMR.EXTEND Leaf

Extend a TDCS.RTMR measurement register.

Table 5.317: TDG.MR.RTMR.EXTEND Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	64B-aligned guest physical address of a 48B extension data buffer		
RDX	Index of the measurement register to be extended		

Table 5.318: TDG.MR.RTMR.EXTEND Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 5.5.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

This function extends one of the RTMR measurement registers in TDCS with the provided extension data in memory.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.319 TDG.MR.RTMR.EXTEND Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Explicit	RCX	GPA	EXTEND_DATA	Blob	R	Private	64B	None
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS.RTMR	SHA384_HASH	N/A	Opaque	N/A	Exclusive
Implicit	N/A	N/A	TDVPR page	TDVPS	None	Opaque	N/A	Shared(i)

In addition to the memory operand checks per the table above, the function checks the following conditions (no specific order is implied):

1. RDX must contain a valid RTMR index.

If successful, the function does the following:

2. Extend the RTMR indexed by RDX with the extension data. Extension is done by calculating SHA384 hash over a 96B buffer, composed as follows:
   - Bytes 0 through 47 contain the current RTMR value.
   - Bytes 48 through 95 contain the extension data.

Completion Status Codes

Table 5.320: TDG.MR.RTMR.EXTEND Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.MR.RTMR.EXTEND is successful.

5.5.7. TDG.MR.VERIFYREPORT

Verify a cryptographic REPORTMACSTRUCT that describes the contents of a TD, to determine that it was created on the current TEE on the current platform.

Table 5.321: TDG.MR.VERIFYREPORT Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	256B-aligned guest physical address of the REPORTMACSTRUCT to be verified.		

Table 5.322: TDG.MR.VERIFYREPORT Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 5.5.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.MR.VERIFYREPORT computes a MAC over the provided REPORTMACSTRUCT structure; it then checks that the computed value is the same as the MAC field of that structure.

Enumeration: Availability of TDG.MR.VERIFYREPORT is enumerated by TDX_FEATURES0.LOCAL_ATTESTATION (bit 8), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.MR.VERIFYREPORT returns a TDX_OPERAND_INVALID(RAX) status.

Retry on Failure: As described in the [Base Spec], there can be cases where report verification fails due to, e.g., microcode update or migration of the reporting TD and the verifying TD to another platform. In such cases it is recommended that a fresh report will be generated by the reporting TD, using TDG.MR.REPORT, and that TDG.MR.VERIFYREPORT will be called again.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.323: TDG.MR.VERIFYREPORT Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Explicit	RCX	GPA	Input report	REPORTMACSTRUCT	R	Private	256B	None
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS.RTMR	SHA384_HASH	N/A	Opaque	N/A	Shared
Implicit	N/A	N/A	TDVPS structure	TDVPS	None	Opaque	N/A	Shared(i)

The function performs the memory operand checks per the table above.

If passed, the function does the following:

1. Calculate MAC over the input REPORTMACSTRUCT fields that are included in the MAC calculation.
2. Compare the calculated MAC to the REPORTMACSTRUCT.MAC field are return a proper status.

Completion Status Codes

Table 5.324: TDG.MR.VERIFYREPORT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INVALID_CPUSVN	See the above note about retrying the operation.
TDX_INVALID_REPORTMACSTRUCT	See the above note about retrying the operation.
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.MR.VERIFYREPORT is successful.

5.5.8. TDG.SERVTD.RD Leaf

As a service TD, read a metadata field (control structure field) of a target TD.

Table 5.325: TDG.SERVTD.RD Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	Binding handle		
RDX	Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored. A value of -1 is a special case: it is not a valid field identifier; in this case the first readable field identifier is returned in RDX.		
R10	Target TD's TD_UUID bits 63:0		
R11	Target TD's TD_UUID bits 127:64		
R12	Target TD's TD_UUID bits 191:128		
R13	Target TD's TD_UUID bits 255:192		

Table 5.326: TDG.SERVTD.RD Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 5.5.1
RDX	RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available. If the input field identifier was -1, RDX returns the first readable field identifier. In case of another error, RDX returns -1.
R8	Contents of the field In case of an error, as indicated by RAX, R8 returns 0.
R10	Updated target TD's TD_UUID bits 63:0 – see the description below.
R11	Updated target TD's TD_UUID bits 127:64 – see the description below.
R12	Updated target TD's TD_UUID bits 191:128 – see the description below.
R13	Updated target TD's TD_UUID bits 255:192 – see the description below.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.SERVTD.RD reads a metadata field (control structure field) of a target TD.

Enumeration: Availability of TDG.SERVTD.RD is enumerated by TDX_FEATURES0.SERVICE_TD (bit 2), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.SERVTD.RD returns a TDX_OPERAND_INVALID(RAX) status.

TD_UUID Update: TD_UUID is updated when the target TD is imported. If the service TD binding to the target TD happened before the target TD was imported, the TD_UUID provided in R13:R10 may no longer be correct. In this case, if the TD_UUID provided in R13:R10 is equal to the pre-import TD_UUID of the target TD, TDG.SERVTD.RD returns TDX_TARGET_UUID_MISMATCH status in RAX, and updates R13:R10 with the imported value of TD_UUID. The called should retry the operation with the new TD_UUID.

Cross-TD Traps: Failure to access the metadata of the target TD may result in a cross-TD trap TD exit to the host VMM. This TD exit is trap like, meaning it happens after TDG.SERVTD.RD has completed its operation. On the following TDH.VP.ENTER, the host VMM may set a HOST_RECOVERABILITY_HINT flag, indicating that TDG.SERVTD.RD may be retried. From the guest TD's perspective, this flag appears in bit 60 of the status code returned in RAX. See the [TDX Module Base Spec] for details.

RDX returns the next host-side readable field identifier. This may be used by the Service TD to dump the target TD metadata readable by the Service TD. To read all the available fields, the service TD can invoke TDG.SERVTD.RD in a loop, starting with field identifier -1 as an input, until RDX returns -1. A status code of TDX_METADATA_FIELD_SKIP indicates that the returned value is not applicable.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.327 TDG.SERVTD.RD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	Target TD's TDR page (from binding handle)	TDR	R	Opaque	N/A	Shared(h)	Shared(h)	Shared(h)
Implicit	N/A	N/A	Service (this) TD's TDR page	TDR	None	Opaque	N/A	Shared(i)	None	None
Implicit	N/A	N/A	Service (this) TD's TDCS structure	TDCS	R	Opaque	N/A	Shared(i)	None	None
Implicit	N/A	N/A	Service (this) TD's TDCS.RTMR	SHA384_ HASH	N/A	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	Service (this) TD's TDVPS structure	TDVPS	None	Opaque	N/A	Shared(i)	Shared(i)	Shared(i)
Implicit	N/A	N/A	Target TD's TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	None	None
Implicit	N/A	N/A	Target TD's TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared(h)	N/A	N/A
Implicit	N/A	N/A	Target TD's Binding table		R	Opaque	N/A	Shared(h)	None	None
Implicit	N/A	N/A	Target TD's TD metadata	N/A	R	Opaque	N/A	None	None	None

If the memory operand checks, per the table above, pass:

1. Based on the provided binding handle and the current (service) TD's TD_UUID, calculate the target TD's TDR HPA and binding slot number.
2. Check that the calculated binding slot number does not exceed target TD's the number of available slots[32].
3. Acquire access to the target TD's TDR in a shared mode.
   3.1. If failed due to HOST_PRIORITY, do a TD exit.
4. Check the target TD state:
   4.1. The target TD's TDR page metadata in PAMT must be correct (PT must be PT_TDR).
   4.2. The target TD is not in a FATAL state (TDR.FATAL is FALSE).
   4.3. The target TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
   4.4. The target TD's TDCS pages must have been allocated (TDR.NUM_TDCX is the required number).

If passed:

5. Check that the target TD's TD_UUID is the same as specified.
   5.1. If failed, and the target TD's PRE_IMPORT_UUID is the same as the specified TD_UUID, abort and return the current target TD's TD_UUID.

If passed:

6. Check that the target TD's binding slot's SERVTD_BINDING_STATE is BOUND.
7. Calculate the current (service) TD's TD_UUID and check it is equal to the target TD's binding slot's SERVTD_UUID.
8. Calculate the current (service) TD's TDINFO_HASH and check it is equal to the target TD's binding slot's SERVTD_TDINFO_HASH.

If passed:

9. Read the control structure field using the algorithm described in 5.3.2.1.

Completion Status Codes

Table 5.328: TDG.SERVTD.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_METADATA_FIELD_ID_INCORRECT	
TDX_METADATA_FIELD_NOT_READABLE	
TDX_METADATA_FIELD_SKIP	Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any.
TDX_METADATA_FIRST_FIELD_ID_IN_CONTEXT	Indicates that the first field ID in context is returned
TDX_METADATA_FIELD_VALUE_NOT_VALID	
TDX_OP_STATE_INCORRECT	This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_OPERAND_ADDR_RANGE_ERROR	This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.

[32] This value is a property of the TDX module and is the same for all TDs.

Completion Status Code	Description
TDX_OPERAND_INVALID	
TDX_PAGE_METADATA_INCORRECT	This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_SERVTD_INFO_HASH_MISMATCH	This service TD5's info hash doesn't match the service TD info hash in the target TD's binding information. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_SERVTD_NOT_BOUND	This service TD is not bound to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_SERVTD_UUID_MISMATCH	This service TD5's TD_UUID doesn't match the service TD UUID in the target TD's binding information. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_SUCCESS	TDG.SERVTD.RD is successful.
TDX_TARGET_UUID_MISMATCH	The target TD's TD_UUID value provided in R13:R10 doesn't match the actual value.
TDX_TARGET_UUID_UPDATED	The target TD's TD_UUID value provided in R13:R10 doesn't match the current actual value, but it does match the TD_UUID that target TD had before it was imported. In this case, the current TD_UUID value is provided in R13:R10, and the operation can be retried.
TDX_TD_FATAL	This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_TD_KEYS_NOT_CONFIGURED	This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_TDCS_NOT_ALLOCATED	This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.

5.5.9. TDG.SERVTD.WR Leaf

As a service TD, write a metadata field (control structure field) of a target TD.

Table 5.329: TDG.SERVTD.WR Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	Binding handle		
RDX	Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored. A value of -1 is a special case: it is not a valid field identifier; in this case the first readable field identifier is returned in RDX.		
R8	Data to write to the field		
R9	A 64b write mask to indicate which bits of the value in R8 are to be written to the field		
R10	Target TD's TD_UUID bits 63:0		
R11	Target TD's TD_UUID bits 127:64		
R12	Target TD's TD_UUID bits 191:128		
R13	Target TD's TD_UUID bits 255:192		

Table 5.330: TDG.SERVTD.WR Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 5.5.1
R8	Previous contents of the field In case of an error, R8 returns 0.
R10	Updated target TD's TD_UUID bits 63:0 – see the description below.
R11	Updated target TD's TD_UUID bits 127:64 – see the description below.
R12	Updated target TD's TD_UUID bits 191:128 – see the description below.
R13	Updated target TD's TD_UUID bits 255:192 – see the description below.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.SERVTD.WR writes a metadata field (control structure field) of a target TD. The value (R8) is written as specified by the write mask (R9). Writing is subject to the field's internal write mask (per the TD's ATTRIBUTES.DEBUG bit). Writing of specific fields is also subject to additional rules.

Table 5.331: Metadata Field Write Rules

Write Mask Bit in R9	Internal Write Mask Bit	Value Bit in R8
0	N/A	Silently ignored
1	0	Must be the same as the current field's bit
1	1	Written to the current field's bit

Enumeration: Availability of TDG.SERVTD.WR is enumerated by TDX_FEATURES0.SERVICE_TD (bit 2), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.SERVTD.WR returns a TDX_OPERAND_INVALID(RAX) status.

TD_UUID Update: TD_UUID is updated when the target TD is imported. If the service TD binding to the target TD happened before the target TD was imported, the TD_UUID provided in R13:R10 may no longer be correct. In this case, if the TD_UUID provided in R13:R10 is equal to the pre-import TD_UUID of the target TD, TDG.SERVTD.WR returns TDX_TARGET_UUID_MISMATCH status in RAX, and updates R13:R10 with the imported value of TD_UUID. The called should retry the operation with the new TD_UUID.

Cross-TD Traps: Failure to access the metadata of the target TD may result in a cross-TD trap TD exit to the host VMM. This TD exit is trap like, meaning it happens after TDG.SERVTD.WR has completed its operation. On the following TDH.VP.ENTER, the host VMM may set a HOST_RECOVERABILITY_HINT flag, indicating that TDG.SERVTD.WR may be retried. From the guest TD's perspective, this flag appears in bit 60 of the status code returned in RAX. See the [TDX Module Base Spec] for details.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.332 TDG.SERVTD.WR Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	Target TD's TDR page (from binding handle)	TDR	RW	Opaque	N/A	Shared(h)	Shared(h)	Shared(h)
Implicit	N/A	N/A	Service (this) TD's TDR page	TDR	None	Opaque	N/A	Shared(i)	None	None
Implicit	N/A	N/A	Service (this) TD's TDCS structure	TDCS	R	Opaque	N/A	Shared(i)	None	None
Implicit	N/A	N/A	Service (this) TD's TDCS.RTMR	SHA384_ HASH	N/A	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	Service (this) TD's TDVPS structure	TDVPS	None	Opaque	N/A	Shared(i)	Shared(i)	Shared(i)
Implicit	N/A	N/A	Target TD's TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)	None	None

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	N/A	Service (this) TD's TDCS.OP_STATE	OP_STATE	R	Opaque	N/A	Shared(h)	N/A	N/A
Implicit	N/A	N/A	Target TD's Binding table		R	Opaque	N/A	Shared(h)	None	None
Implicit	N/A	N/A	Target TD's TD metadata	N/A	RW	Opaque	N/A	None	None	None

If the memory operand checks, per the table above, pass:

1. Based on the provided binding handle and the current (service) TD's TD_UUID, calculate the target TD's TDR HPA and binding slot number.
2. Check that the calculated binding slot number does not exceed target TD's the number of available slots[33].
3. Acquire access to the target TD's TDR in a shared mode.
   3.1. If failed due to HOST_PRIORITY, do a TD exit.
4. Check the target TD state:
   4.1. The target TD's TDR page metadata in PAMT must be correct (PT must be PT_TDR).
   4.2. The target TD is not in a FATAL state (TDR.FATAL is FALSE).
   4.3. The target TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
   4.4. The target TD's TDCS pages must have been allocated (TDR.NUM_TDCX is the required number).
   4.5. The target TD has not been paused for export.

If passed:

5. Check that the target TD's TD_UUID is the same as specified.
   5.1. If failed, and the target TD's PRE_IMPORT_UUID is the same as the specified TD_UUID, abort and return the current target TD's TD_UUID.

If passed:

6. Check that the target TD's binding slot's SERVTD_BINDING_STATE is BOUND.
7. Calculate the current (service) TD's TD_UUID and check it is equal to the target TD's binding slot's SERVTD_UUID.
8. Calculate the current (service) TD's TDINFO_HASH and check it is equal to the target TD's binding slot's SERVTD_TDINFO_HASH.

If passed:

9. Write the control structure field and return its old value, using the algorithm described in 5.3.2.2.

Completion Status Codes

Table 5.333: TDG.SERVTD.WR Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_METADATA_FIELD_ID_INCORRECT	
TDX_METADATA_FIELD_NOT_READABLE	
TDX_METADATA_FIELD_NOT_WRITABLE	
TDX_METADATA_FIELD_VALUE_NOT_VALID	
TDX_METADATA_WR_MASK_NOT_VALID	

[33] This value is a property of the TDX module and is the same for all TDs.

Completion Status Code	Description
TDX_OP_STATE_INCORRECT	This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_OPERAND_ADDR_RANGE_ERROR	This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_PAGE_METADATA_INCORRECT	This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_SERVTD_INFO_HASH_MISMATCH	This service TD5's info hash doesn't match the service TD info hash in the target TD's binding information. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_SERVTD_NOT_BOUND	This service TD is not bound to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_SERVTD_UUID_MISMATCH	This service TD5's TD_UUID doesn't match the service TD UUID in the target TD's binding information. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_SUCCESS	TDG.SERVTD.WR is successful.
TDX_TARGET_UUID_MISMATCH	The target TD's TD_UUID value provided in R13:R10 doesn't match the actual value.
TDX_TARGET_UUID_UPDATED	The target TD's TD_UUID value provided in R13:R10 doesn't match the current actual value, but it does match the TD_UUID that target TD had before it was imported. In this case, the current TD_UUID value is provided in R13:R10, and the operation can be retried.
TDX_TD_FATAL	This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.
TDX_TD_KEYS_NOT_CONFIGURED	This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.

Completion Status Code	Description
TDX_TDCS_NOT_ALLOCATED	This status code refers to the target TD. Bit 60 (HOST_RECOVERABILITY_HINT) contains a hint from the host VMM that the error condition has been resolved and the service TD can retry the operation.

5.5.10. TDG.SYS.RD Leaf

Read a TDX Module global-scope metadata field.

Table 5.334: TDG.SYS.RD Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RDX	Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored. A value of -1 is a special case: it is not a valid field identifier; in this case the first readable field identifier is returned in RDX.		

Table 5.335: TDG.SYS.RD Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 5.5.1
RDX	If the input field identifier was -1, RDX returns the first readable field identifier. Else, in case of an error, RDX returns -1. On success, RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available.
R8	Contents of the field In case of no success, as indicated by RAX, R8 returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.SYS.RD reads a TDX Module global-scope metadata field.

RDX returns the next guest-side readable field identifier. This may be used by the guest TD to enumerate the TDX Module's capabilities and configuration. To read all the available fields, the guest TD can invoke TDG.SYS.RD in a loop, starting with field identifier -1 as an input, until RDX returns -1. A status code of TDX_METADATA_FIELD_SKIP indicates that the returned value is not applicable. Alternatively, the guest TD can use TDG.SYS.RDALL.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.336 TDG.SYS.RD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDVPS structure	TDVPS	None	Opaque	N/A	Shared(i)

1. Read the requested field using the algorithm described in 5.3.2.1.
2. Return the next readable field identifier, or a value of 0 if none exists.
3. Return the field value.

Completion Status Codes

Table 5.337: TDG.SYS.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_METADATA_FIELD_ID_INCORRECT	
TDX_METADATA_FIELD_NOT_READABLE	
TDX_METADATA_FIELD_SKIP	Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any.
TDX_METADATA_FIRST_FIELD_ID_IN_CONTEXT	Indicates that the first field ID in context is returned
TDX_METADATA_FIELD_SKIP	Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any.
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.SYS.RD is successful.

5.5.11. TDG.SYS.RDALL Leaf

Read all guest-readable TDX module global-scope metadata fields.

Table 5.338: TDG.SYS.RDALL Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RDX	The GPA of a 4KB buffer where a metadata list will be returned The buffer must be aligned on 4KB. In case of error, some field value entries might not contain valid data.		
R8	Initial field identifier – see 3.10 If R8's value is -1, then TDG.SYS.RDALL will start from the first global-scope metadata field identifier. Else, LAST_ELEMENT_IN_FIELD, LAST_FIELD_IN_SEQUENCE, WRITE_MASK_VALID and CONTEXT_CODE fields are ignored. The FIELD_CODE must be the code of the first element of a metadata field.		

Table 5.339: TDG.SYS.RDALL Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 5.5.1
R8	Next field identifier. A value of -1 means all applicable field identifiers have been returned in the metadata list. In case of an error, as indicated by RAX, R8 returns -1.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.SYS.RDALL reads all host-readable TDX Module global-scope metadata fields into a metadata list in the provided page.

If one or more applicable fields do not fit in the provided list buffer, the function can be invoked in a loop, each invocation providing an initial field identifier returned as the next field identifier of the previous invocation, as shown in the following example:

1. NEXT_FIELD_ID = -1
2. Repeat:
   2.1. Set LIST_BUFFER to the next 4K buffer
   2.2. Invoke TDG.SYS.RDALL(RDX = LIST_BUFFER, RDX = NEXT_FIELD_ID)
   2.3. STATUS = RAX, NEXT_FIELD_ID = R8

   Until ((STATUS is a non-recoverable error) or (NEXT_FIELD_ID is -1))

The function never returns an empty list if there's no error.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.340: TDG.SYS.RDALL Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Explicit	RDX	GPA	Metadata List	MD_LIST	RW	Private	4096	None
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDVPS structure	TDVPS	None	Opaque	N/A	Shared(i)

If the memory operand checks, per the table above, pass:

1. Dump all guest-readable metadata fields into the provided list buffer.

Completion Status Codes

Table 5.341: TDG.SYS.RDALL Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.SYS.RDALL is successful.

5.5.12. TDG.VM.RD Leaf

Read a TD-scope metadata field (control structure field) of a TD.

Table 5.342: TDG.VM.RD Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Version number may be 0 or 1. See the enumeration details below.
	63:24	Reserved	Must be 0
RCX	Reserved, must be 0		
RDX	Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored. For TDG.VM.RD version 1 or higher, a value of -1 is a special case: it is not a valid field identifier; in this case the first readable field identifier is returned in RDX.		

Table 5.343: TDG.VM.RD Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 5.5.1
RDX	For TDG.VM.RD version 0, RDX is unmodified. For TDG.VM.RD version 1 or higher: • If the input field identifier was -1, RDX returns the first readable field identifier. • Else, in case of an error, RDX returns -1. On success, RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available.
R8	Contents of the field In case of no success, as indicated by RAX, R8 returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.VM.RD reads a VM-scope metadata field (control structure field) of a TD.

Enumeration: Availability of TDG.VM.RD version 1 is enumerated by TDX_FEATURES0.ENHANCED_METADATA (bit 3), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.VM.RD with a version number higher than 0 returns a TDX_OPERAND_INVALID(RAX) status.

If version 1 or higher is specified in RAX, RDX returns the next host-side readable field identifier. This may be used by the guest TD to dump the guest readable TD metadata. To read all the available fields, the guest TD can invoke TDG.VM.RD

in a loop, starting with field identifier -1 as an input, until RDX returns -1. A status code of TDX_METADATA_FIELD_SKIP indicates that the returned value is not applicable.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.344 TDG.VM.RD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDVPS structure	TDVPS	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TD metadata (guest-side access)	N/A	R	Opaque	N/A	Shared

If the memory operand checks, per the table above, pass:

10. Read the control structure field using the algorithm described in 5.3.2.1.

Completion Status Codes

Table 5.345: TDG.VM.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_METADATA_FIELD_ID_INCORRECT	
TDX_METADATA_FIELD_NOT_READABLE	
TDX_METADATA_FIELD_SKIP	Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any.
TDX_METADATA_FIRST_FIELD_ID_IN_CONTEXT	Indicates that the first field ID in context is returned
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.VM.RD is successful.

5.5.13. TDG.VM.WR Leaf

Write a TD-scope metadata field (control structure field) of a TD.

Table 5.346: TDG.VM.WR Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	Reserved, must be 0		
RDX	Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored.		
R8	Data to write to the field		
R9	A 64b write mask to indicate which bits of the value in R8 are to be written to the field		

Table 5.347: TDG.VM.WR Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 5.5.1
R8	Previous contents of the field In case of an error, as indicated by RAX, R8 returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.VM.WR writes a VM-scope metadata field (control structure field) of a TD. The value (R8) is written as specified by the write mask (R9). Writing is subject to the field's internal write mask (per the TD's ATTRIBUTES.DEBUG bit). Writing of specific fields is also subject to additional rules.

Table 5.348: Metadata Field Write Rules

Write Mask Bit in R9	Internal Write Mask Bit	Value Bit in R8
0	N/A	Silently ignored
1	0	Must be the same as the current field's bit
1	1	Written to the current field's bit

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.349 TDG.VM.WR Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDVPS structure	TDVPS	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TD metadata (guest-side access)	N/A	R	Opaque	N/A	Shared

If the memory operand checks, per the table above, pass:

1. Write the control structure field and return its old value, using the algorithm described in 5.3.2.2.

Completion Status Codes

Table 5.350: TDG.VM.WR Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_METADATA_FIELD_ID_INCORRECT	
TDX_METADATA_FIELD_NOT_READABLE	
TDX_METADATA_FIELD_NOT_WRITABLE	
TDX_METADATA_FIELD_VALUE_NOT_VALID	
TDX_METADATA_WR_MASK_NOT_VALID	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.VM.WR is successful.

5.5.14. TDG.VP.CPUIDVE.SET Leaf

TDG.VP.CPUIDVE.SET controls unconditional #VE on CPUID execution by the guest TD.

Note: TDG.VP.CPUIDVE.SET is provided for backward compatibility. The guest TD may control the same settings by writing to the VCPU-scope metadata fields CPUID_SUPERVISOR_VE and CPUID_USER_VE using TDG.VP.WR.

Table 5.351: TDG.VP.CPUIDVE.SET Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	Controls whether CPUID executed by the guest TD will cause #VE(CONFIG_PARAVIRT) unconditionally		
	Bits	**Name**	**Description**
	0	SUPERVISOR	Flags that when CPL is 0, a CPUID executed by the guest TD will cause a #VE(CONFIG_PARAVIRT) unconditionally
	1	USER	Flags that when CPL > 0, a CPUID executed by the guest TD will cause a #VE(CONFIG_PARAVIRT) unconditionally
	63:2	RESERVED	Reserved: must be 0

Table 5.352: TDG.VP.CPUIDVE.SET Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 5.5.1
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

This function controls whether execution of CPUID by the guest TD, when running in supervisor mode and/or in user mode, will unconditionally result in a #VE(CONFIG_PARAVIRT).

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.353 TDG.VP.CPUIDVE.SET Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Implicit	N/A	N/A	TDVPS structure	TDVPS	RW	Opaque	N/A	Shared(i)

In addition to the memory operand checks per the table above, the function checks the following conditions (no specific order is implied):

1. Reserved bits of RCX must be 0.

If successful, the function does the following:

2. Update the TDVPS.CPUID_VE flags which control unconditional #VE(CONFIG_PARAVIRT) injection for CPUID for the current VCPU.

Completion Status Codes

Table 5.354: TDG.VP.CPUIDVE.SET Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.VP.CPUIDVE.SET is successful.

5.5.15. TDG.VP.ENTER Leaf

Enter L2 VCPU operation.

From the L1 VMM's point of view, TDG.VP.ENTER is a complex operation that normally involves L1→L2 VM entry and L2→L1 VM exit; however, it may fail before L2 VM entry and may also involve TD exits and entries. Therefore, output operands are specified by multiple tables below.

Inputs

Table 5.355: TDG.VP.ENTER Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	TDCALL instruction leaf number and version, see 5.5.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the TDCALL interface function
		23:16	Version Number	Selects the TDCALL interface function version Must be 0
		63:24	Reserved	Must be 0
RCX	VM_FLAGS	VM identifier and flags		
		Bits	**Name**	**Description**
		1:0	INVD_TRANSLATIONS	Controls how TDG.VP.ENTER flushes the TLB context and extended paging structure (EPxE) caches associated with the L2 VM before entering the L2 VCPU
		51:2	Reserved	Reserved: must be 0
		53:52	VM	L2 virtual machine index (must be 1 or higher)
		63:54	Reserved	Reserved: must be 0
RDX	GUEST_STATE_GPA	The GPA of a 256-bytes aligned L2_ENTER_GUEST_STATE structure - see 3.8.1 for details.		

Table 5.356: INVD_TRANSLATION Definition

Value	Address Translation Invalidation	Underlying Mechanism	Comments
0	No invalidation	None	
1	Invalidate all TLB entries and extended paging-structure translations (EPxE) associated with the L2 VM being entered	INVEPT single-context invalidation (type 1)	
2	Invalidate all TLB entries associated with the L2 VM being entered	INVVPID single-context invalidation (type 1)	See enumeration details below.
3	Invalidate TLB entries associated with the L2 VM being entered, excluding global translations	INVVPID single-context invalidation, retaining global translations (type 3)	See enumeration details below.

Outputs

TDG.VP.ENTER output format depends on how the function was terminated.

The following table details TDG.VP.ENTER output operands when the interface function returns **without entering the L2 VCPU** due to an error or some other condition.

Table 5.357: TDG.VP.ENTER Output Operands Definition on No L2 VM Entry

Operand	Name	Description		
RAX	Status	SEAMCALL instruction return code		
		Bit(s)	**Name**	**Description**
		47:32	CLASS and DETAILS_L1	May have the following values: • TDX_PENDING_INTERRUPT, indicating that an interrupt is pending for L1 • Any other value not in the table below
		Other		See the function completion status definition in 5.5.1
Other		Unmodified		

The following table details TDG.VP.ENTER output operands when L2 VM entry succeeds, and later an L2 VM exit occurs due to a **VMX architectural exit reason**.

Table 5.358: TDG.VP.ENTER Output Operands Definition on an L2→L1 Exits Following an L1→L2 Entry

Operand	Name	Description		
RAX	Status	SEAMCALL instruction return code		
		Bit(s)	**Name**	**Description**
		31:0	DETAILS_L2: Exit Reason	L2 VMCS exit reason
		47:32	CLASS and DETAILS_L1	May have the following values: • TDX_SUCCESS, indicating a normal L2→L1 exit • TDX_L2_EXIT_PENDING_INTERRUPT, indicating an L2→L1 exit due to an interrupt posted to L1 • TDX_L2_EXIT_HOST_ROUTED_*, indicating a TD exit from L2 where the host VMM requested resumption of L1 Other values in the range 0x1100 through 0x111F are reserved for future additional status codes that indicate an L2→L1 exit following an L1→L2 entry.
		Other		See the function completion status definition in 5.5.1
RCX	Exit Qualification	exit qualification from L2 VMCS		
RDX	Guest Linear Address	guest-linear address from L2 VMCS		
RSI	CS Info	CS selector, AR and limit		
		Bits	**Details**	
		15:0	CS Selector	
		31:16	CS AR bit 15:0	
		63:32	CS Limit	

Operand	Name	Description			
RDI	CS Base	CS base address			
R8	Guest Physical Address	guest-physical address from L2 VMCS			
R9	VM-Exit Interruption Information	The following information is provided for L2 VM exits due to vectored events. In other cases, R9 content should be ignored.			
		Bits	**Details**		
		31:0	VM-Exit Interruption Information		
		63:32	VM-Exit Interruption Error Code		
R10	IDT-Vectoring Information	The following information is provided for L2 VM exits that occur during event delivery. In other cases, R10 content should be ignored.			
		Bits	**Details**		
		31:0	IDT-Vectoring Information		
		63:32	IDT-Vectoring Error Code		
R11	VM-Exit Instruction Information	The following information is provided for L2 VM exits due to instruction execution. In other cases, R11 content should be ignored.			
		Bits	**Details**		
		31:0	VM-Exit Instruction Information		
		63:32	VM-Exit Instruction Length		
R12	Additional Exit Information	Additional exit information			
		Bits	**Details**		
		1:0	CPL (from GuestSS.AR.DPL)		
		63:2	Reserved, cleared to 0		
R13	Extended Exit Qualification	Extended exit qualification			
		3:0	Extended exit qualification type		
			Value	**Name**	**Description**
			0	NONE	No extended exit qualification
			6	PENDING_EPT_VIOLATION	Extended exit qualification for an EPT violation due to L2 VM access to a PENDING page
			Other	Reserved	
		63:4	Reserved, set to 0		
R14	VM-Exit Extended Instruction Information	If both the TDX module and the CPU support Intel® APX (Advanced Performance Extensions), as enumerated by TDX_FEATURES0.APX (bit 28), then R14 returns the L2 VMCS' VM-Exit Extended Instruction Information field. Else, R14 returns 0.			
RBX, R15	None	Cleared to 0			
Other state	Any state that the L2 VM is allowed to use may be modified.				

CPU State Preservation Following a Successful L1→L2 VM Entry and an L2→L1 VM Exit

Following a successful L1→L2 VM entry and an L2→L1 VM exit, some CPU state is modified:

- General purpose register (GPR) values are not preserved.
- Any state that the L2 VM is allowed to use may be modified.

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.VP.ENTER transitions the VCPU into L2 VM operation. The function returns either if failed to enter L2 VM or after successful entry to L2 VM and then exit from L2 VM.

Enumeration: Availability of TDG.VP.ENTER is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDG.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.VP.ENTER returns a TDX_OPERAND_INVALID(RAX) status.

Available INVD_TRANSLATION values are enumerated by TDX_FEATURES0.L2_TLB_INVD_OPT (bit 19).

Availability of PENDING_EPT_VIOLATION indication in R13 is enumerated by TDX_FEATURES.PENDING_EPT_VIOLATION_V2 (bit 16).

The following table lists non-error TDG.VP.ENTER termination conditions:

Table 5.359: TDG.VP.ENTER Termination Cases

Case	Status in RAX[63:32]	Description
Normal	TDX_SUCCESS	L1→L2 entry was successful, followed by an L2→L1 exit. L2 VM exit information is provided in output GPRs.
Host Requested L2 Exit	TDX_L2_EXIT_HOST_ ROUTED_ASYNC	L1→L2 entry was successful. Later, following direct TD exit from L2, the host VMM requested resumption of L1. L2 CPU state is updated in the register list. L2 VM exit information is provided in output GPRs. This information was provided to the host VMM on TD exit; it may or may not be meaningful to the L1 VMM. In case of TDG.VP.VMCALL, the L2 CPU state is the state after completion of that function, e.g., GPR values are as returned by the host VMM as inputs to TDH.VP.ENTER. This condition is sticky. I.e., if resumption of L1 encountered a problem that required a TD exit (e.g., an EPT violation) the following TD entry resumes L1 and provides the same TDX_L2_EXIT_HOST_ROUTED status.
Host Requested L2 Exit following TDG.VP.VMCALL	TDX_L2_EXIT_HOST_ ROUTED_TDVMCALL	L1→L2 entry was successful. Later, following TDG.VP.VMCALL that caused a direct TD exit from L2, the host VMM requested resumption of L1. L2 CPU state is updated in the L2_ENTER_GUEST_STATE structure. The L2 CPU state is the state after completion of TDG.VP.VMCALL, e.g., GPR values are as returned by the host VMM as inputs to TDH.VP.ENTER. L2 VM exit information is provided in output GPRs. This information was provided to the host VMM on the last TD exit; it may or may not be meaningful to the L1 VMM. This condition is sticky. I.e., if resumption of L1 encountered a problem that required a TD exit (e.g., an EPT violation) the following TD entry resumes L1 and provides the same TDX_L2_EXIT_HOST_ROUTED_TDVMCALL status. Note that in such case the L2 VM exit information reflects, e.g., the EPT violation, not the original TDG.VP.VMCALL exit.

Case	Status in RAX[63:32]	Description
Pending Interrupt L2 Exit	TDX_L2_EXIT_PENDING_ INTERRUPT	L1→L2 entry was successful. Later, an L2→L1 exit happened due to an interrupt that was posted to L1. L2 CPU state is updated in the register list. L2 VM exit information is provided in output GPRs but is not necessarily meaningful to the L1 VMM (e.g., VM-exit interruption information is for the external interrupt that triggered the L2→L1 exit).
No L2 Entry	Other	L1→L2 entry was aborted because of some condition, which may or may not be an error, as indicated by RAX bit 63. E.g., entry may have been aborted due to a pending virtual interrupt. In this case, the L1 VMM typically sets RFLAGS.IF to 1, handles the interrupt and then invokes TDG.VP.ENTER again.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.360: TDG.VP.ENTER Memory Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Explicit	RDX	GPA	Guest state	L2_ENTER_GUEST_ STATE	RW	Private	256B	None
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDVPS structure	TDVPS	None	Opaque	N/A	Shared(i)

In addition to the explicit memory operand checks per the table above, the function checks the following conditions:

1. VM_FLAGS:
   1.1. VM index must be between 1 and TDCS.NUM_L2_VMS.
   1.2. The reserved fields must be 0.
2. GUEST_STATE_GPA is a valid private GPA.

If passed:

3. Check if there is a pending virtual interrupt to L1. This is indicated by RVI[7:4] > VPPR[7:4]. If so, terminate with a TDX_PENDING_INTERRUPT status.

If no interrupt is pending:

4. Translate GPAs:
   4.1. Translate TDG.VP.ENTER memory output operands GPAs. Translation needs to be done in one of the following cases:
   - The GPA is different than stored in TDVPS from last time TDG.VP.ENTER was called for this VM.
   - The GPA has been blocked since last translated.
   - HPA shadow for this GPA is NULL_PA.

   Translation failure leads to an EPT violation TD exit.

   If passed:

   4.2. Translate GPAs of L2 VMCS fields. Translation needs to be done in one of the following cases:
   - The GPA has been blocked since last translated.
   - HPA shadow for this GPA is NULL_PA.

If passed:

5. If INVD_TRANSLATIONS is not 0:
   5.1. Execute INVEPT type 1, INVVPID type 1 or INVVPID type 3 to flush address translations of the L2 VM.

6. Read the provided register list
   6.1. Write VMCS-stored register values (e.g., RSP) to the L2 VMCS.
   6.2. Load GPR values to the CPU's GPRs.
7. Execute VMLAUNCH or VMRESUME depending on whether the entered VCPU and L2 VM (i.e., the current L2 VMCS) has been launched on this LP since the VCPU's last association with the LP (TDVPS.LAUNCHED[VM]).

Completion Status Codes

Table 5.361: TDG.VP.ENTER Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_L2_EXIT_HOST_ROUTED_ASYNC	L1→L2 entry succeeded. Later, following an asynchronous TD exit from L2, the host VMM requested resumption of L1.
TDX_L2_EXIT_HOST_ROUTED_TDVMCALL	L1→L2 entry succeeded. Later, following a TDG.VP.VMCALL TD exit from L2, the host VMM requested resumption of L1.
TDX_L2_EXIT_PENDING_INTERRUPT	L1→L2 entry succeeded, and later L2→L1 exit happened due to an interrupt that was posted to L1 and is pending.
TDX_OPERAND_INVALID	
TDX_PENDING_INTERRUPT	L1→L2 entry was aborted because an interrupt is pending for the L1 VMM. This indication is returned even if the L1 VMMs cleared RFLAGS.IF.
TDX_SUCCESS	

5.5.16. TDG.VP.INFO Leaf

Get guest TD execution environment information.

Table 5.362: TDG.VP.INFO Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0

Table 5.363: TDG.VP.INFO Output Operands Definition

Operand	Description		
RAX	TDCALL instruction return code – see 5.5.1 – returns a constant value of TDX_SUCCESS (0)		
RCX	**Bits**	**Name**	**Description**
	5:0	GPAW	The effective GPA width (in bits) for this TD (do not confuse with MAXPA). SHARED bit is at GPA bit GPAW-1. Only GPAW values 48 and 52 are possible.
	63:6	RESERVED	Reserved: 0
RDX	The TD's ATTRIBUTES (provided as input to TDH.MNG.INIT)		
R8	**Bits**	**Name**	**Description**
	31:0	NUM_VCPUS	Number of Virtual CPUs that are usable (i.e., either active or ready)
	63:32	MAX_VCPUS	TD's maximum number of Virtual CPUs (provided as input to TDH.MNG.INIT)
R9	**Bits**	**Name**	**Description**
	31:0	VCPU_INDEX	Virtual CPU index, starting from 0 and allocated sequentially on each successful TDH.VP.INIT
	63:32	RESERVED	Reserved for enumerating future Intel TDX module capabilities, etc.: set to 0
R10	**Bits**	**Name**	**Description**
	0	SYS_RD	Indicates that the TDG.SYS.RD/RDM/RDALL functions are available. Further enumeration can be done using these functions.
	63:1	RESERVED	Reserved – set to 0
R11	Reserved for enumerating future Intel TDX module capabilities, etc.: set to 0		
Other	Unmodified		

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.VP.INFO provides the TD software with execution environment information – beyond information that is provided by CPUID.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.364: TDG.VP.INFO Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDVPS structure	TDVPS	R	Opaque	N/A	Shared(i)

Completion Status Codes

Table 5.365: TDG.VP.INFO Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_SUCCESS	TDG.VP.INFO is successful.

5.5.17. TDG.VP.INVEPT Leaf

Invalidate cached EPT translations for selected L2 VMs.

Table 5.366: TDG.VP.INVEPT Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	VM index bitmap Bit N value of 1 indicates a request to invalidate EPT for VM index N. N must be between 1 and the number of L2 VMs in this TD.		
	Bits	**Name**	**Description**
	0	Reserved	Reserved: must be 0
	1	L2_VM_1	Invalidate EPT for L2 VM #1
	2	L2_VM_2	Invalidate EPT for L2 VM #2
	3	L2_VM_3	Invalidate EPT for L2 VM #3
	63:4	Reserved	Reserved: must be 0

Table 5.367: TDG.VP.INVEPT Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.VP.INVEPT executes INVEPT to invalidate the EPT translations of the specified L2 VMs.

Enumeration: Availability of TDG.VP.INVEPT is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.VP.INVEPT returns a TDX_OPERAND_INVALID(RAX) status.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.368 TDG.VP.RD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDVPS structure	TDVPS	R	Opaque	N/A	Shared(i)

If the memory operand checks, per the table above, pass:

1. Check the validity of the VM index bit mask
2. Execute INVEPT type 1 (single context invalidation) for the specified L2 VMs' EPTP.

Completion Status Codes

Table 5.369: TDG.VP.INVEPT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.VP.INVEPT is successful.

5.5.18. TDG.VP.INVGLA Leaf

Invalidate Guest Linear Address (GLA) mappings in the translation lookaside buffers (TLBs) and paging-structure caches for a specified L2 VM and a specified list of 4KB-aligned linear addresses.

Table 5.370: TDG.VP.INVGLA Input Operands Definition

Operand	Name	Description		
RAX	Leaf and Version	TDCALL instruction leaf number and version, see 5.5.1		
		Bits	**Field**	**Description**
		15:0	Leaf Number	Selects the TDCALL interface function
		23:16	Version Number	Selects the TDCALL interface function version Must be 0
		63:24	Reserved	Must be 0
RCX	VM_AND_FLAGS	VM identifier and flags		
		Bits	**Name**	**Description**
		0	LIST	0: RDX contains a single GLA list entry 1: RDX contains the GPA and other information of a GLA list in memory.
		51:1	Reserved	Reserved: must be 0
		53:52	VM	L2 virtual machine index (must be 1 or higher)
		63:54	Reserved	Reserved: must be 0
RDX	GLA_LIST_ENTRY or GLA_LIST_INFO	Depending on the LIST flag in RCX, RDX contains either of the following: • A single GLA_LIST_ENTRY, specifying up to 512 consecutive guest linear addresses, each aligned on 4KB. • GLA_LIST_INFO, specifying the GPA of a guest linear address (GLA) list in private memory. Each entry in the GLA list specifies up to 512 consecutive guest linear addresses, each aligned on 4KB. GLA_LIST_INFO also specifies the first and last GLA list entries to process. See 3.6.4 for details.		

5

Table 5.371: TDG.VP.INVGLA Output Operands Definition

Operand		Description
RAX	Status	TDCALL instruction return code
RDX	GLA_LIST_ENTRY or GLA_LIST_INFO	Depending on the LIST flag provided as input in RCX, RDX contains either of the following: • If LIST was 0, RDX contains the single GLA_LIST_ENTRY provided as an input, unmodified. • If LIST was 1, RDX contains the GLA_LIST_INFO provided as input, but with the FIRST_ENTRY and NUM_ENTRIES fields updated to reflect the number of entries processed so far. If all entries have been processed successfully, NUM_ENTRIES is set to 0.
Other		Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.VP.INVGLA executes INVVPID type 0 to invalidate the cached translations for the specified list of 4KB page Guest Linear Addresses (GLA) of the specified L2 VM.

Enumeration: Availability of TDG.VP.INVGLA is enumerated by TDX_FEATURES0.TD_PARTITIONING (bit 7), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.VP.INVGLA returns a TDX_OPERAND_INVALID(RAX) status.

Interruptibility If, during its execution, TDG.VP.INVGLA detects that an external interrupt is pending, it may resume the guest TD with the CPU state unmodified, except for the following:

- GLA_LIST_INFO in RDX is updated to reflect the GLAs processed so far.

This allows the external interrupt to be recognized, causing a VM exit or a posted interrupt delivery. Typically, TDG.VP.INVGLA will be re-invoked (since RIP has not changed) and continue its work. Guest TD software is not directly involved.

Guest TD software is not directly involved. Guest TD should not precede the TDCALL with an STI instruction or a MOV to SS instruction. Posted interrupts may be delivered when the TDCALL flow is interrupted.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.372 TDG.VP.INVGLA Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Explicit	RDX	GPA	GLA list page	GLA_LIST	R	Private	4096	None
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDVPS structure	TDVPS	R	Opaque	N/A	Shared(i)

If the memory operand checks, per the table above, pass, the function checks the following conditions (no specific order is implied):

1. VM is the index of an existing L2 VM.

If passed:

2. If the LIST flag in RCX is 0, process a single GLA_LIST_ENTRY in RDX:
   2.1. With current GLA starting from BASE_GLA, repeat for each page through LAST_PAGE:
      2.1.1. Execute INVVPID type 0, providing the current GLA and the VM's VPID.
      2.1.2. Advance the current GLA by 4KB.
3. Else (LIST flag in RCX is 1), process a GLA list:
   3.1. Check the validity of GLA_LIST_INFO.

   If passed:

   3.2. Translate the GPA list's GPA.
      3.2.1. On translation error, do a TD exit with an EPT violation indication.
   3.3. Repeat for NUM_ENTRIES from FIRST_ENTRY:
      3.3.1. Read the current GLA_LIST_ENTRY.
      3.3.2. Process the current entry as described in the single-entry case above.
      3.3.3. If there is a pending interrupt and this was not the last entry:
         3.3.3.1. Update RDX to reflect the work done so far: set FIRST_ENTRY to the index of the next entry and NUM_ENTRIES to the remaining number of entries.
         3.3.3.2. Resume the guest TD without updating RIP or any other state except for RDX.

Completion Status Codes

Table 5.373: TDG.VP.INVGLA Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.VP.INVGLA is successful.

Intel TDX Application Binary Interface (ABI) Reference

5.5.19. TDG.VP.RD Leaf

Read a VCPU-scope metadata field (control structure field) of a TD.

Table 5.374: TDG.VP.RD Input Operands

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	Reserved, must be 0		
RDX	Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and ELEMENT_SIZE_CODE components of the field identifier are ignored. A value of -1 is a special case: it is not a valid field identifier; in this case the first readable field identifier is returned in RDX.		

Table 5.375: TDG.VP.RD Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 5.5.1
RDX	If the input field identifier was -1, RDX returns the first readable field identifier. Else, in case of an error, RDX returns -1. On success, RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available.
R8	Contents of the field In case of no success, as indicated by RAX, R8 returns 0.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.VP.RD reads a VCPU-scope metadata field (control structure field) of a TD.

RDX returns the next host-side readable field identifier. This may be used by the guest TD to dump the guest readable VCPU metadata. To read all the available fields, the guest TD can invoke TDG.VP.RD in a loop, starting with field identifier -1 as an input, until RDX returns -1. A status code of TDX_METADATA_FIELD_SKIP indicates that the returned value is not applicable.

Enumeration: Availability of TDG.VP.RD enumerated by TDX_FEATURES0.ENHANCED_METADATA (bit 3), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.VP.RD returns a TDX_OPERAND_INVALID(RAX) status.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.376 TDG.VP.RD Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDVPS structure	TDVPS	R	Opaque	N/A	Shared(i)

If the memory operand checks, per the table above, pass:

1. Read the control structure field using the algorithm described in 5.3.2.1.

Completion Status Codes

Table 5.377: TDG.VP.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_METADATA_FIELD_ID_INCORRECT	
TDX_METADATA_FIELD_NOT_READABLE	
TDX_METADATA_FIELD_SKIP	Indicates that the field value being read is not applicable and needs to be skipped. If called in a loop, use RDX as the identifier of the next field to be read, if any.
TDX_METADATA_FIRST_FIELD_ID_IN_CONTEXT	Indicates that the first field ID in context is returned
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.VP.RD is successful.

5.5.20. TDG.VP.VEINFO.GET Leaf

Intel SDM, Vol. 3, 24.9.4 Information for VM Exits Due to Instruction Execution
Intel SDM, Vol. 3, 25.5.6 Virtualization Exceptions
Intel SDM, Vol. 3, 27.2.5 Information for VM Exits Due to Instruction Execution

Get Virtualization Exception Information for the recent #VE exception.

Table 5.378: TDG.VP.VEINFO.GET Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Version 0 is always supported. Version 1 is supported if the TDX module enumerates TDX_FEATURES0.VE_REDUCTION (bit 30) as 1. Version 2 is supported if both the TDX module and the CPU support Intel® APX, as enumerated by TDX_FEATURES0.APX (bit 28).
	63:24	Reserved	Must be 0

Table 5.379: TDG.VP.VEINFO.GET Output Operands Definition

Operand	Description		
RAX	TDCALL instruction return code – see 5.5.1		
RCX	**Bits**	**Name**	**Description**
	31:0	Exit Reason	The 32-bit value that would have been saved into the VMCS as an exit reason if a VM exit had occurred instead of the virtualization exception
	39:32	#VE Category	If TDG.VP.VEINFO.GET was called with version 0, this field returns 0. Else, this field returns the #VE category, as defined in the [Base FAS].
	63:40	Reserved	Reserved: 0
	In case of an error, RCX returns 0.		
RDX	Exit Qualification: the 64-bit value that would have been saved into the VMCS as an exit qualification if a legacy VM exit had occurred instead of the virtualization exception In case of an error, RDX returns 0.		
R8	Guest Linear Address: the 64-bit value that would have been saved into the VMCS as a guest-linear address if a legacy VM exit had occurred instead of the virtualization exception In case of an error, R8 returns 0.		
R9	Guest Physical Address: the 64-bit value that would have been saved into the VMCS as a guest-physical address if a legacy VM exit had occurred instead of the virtualization exception In case of an error, R9 returns 0.		
R10	**Bits**	**Name**	**Description**

Operand	Description		
	31:0	VM-exit instruction length	The 32-bit value that would have been saved into the VMCS as VM-exit instruction length if a legacy VM exit had occurred instead of the virtualization exception
	63:32	VM-exit instruction information	The 32-bit value that would have been saved into the VMCS as VM-exit instruction information if a legacy VM exit had occurred instead of the virtualization exception
	The content of R10 is only applicable for TDX-extended #VE (injected by the TDX module), where Exit Reason is not EPT violation (48). It should be ignored for EPT violations converted by the CPU to #VE. In case of an error, R10 returns 0.		
R11	If TDG.VP.VEINFO.GET was called with version 0, then R11 is unmodified. Else: • If both the TDX module and the CPU support Intel® APX, as enumerated by TDX_FEATURES0.APX (bit 28), and there was no error, then R11 returns Extended Instruction Information: the 64-bit value that would have been saved into the VMCS as an extended instruction information if a legacy VM exit had occurred instead of the virtualization exception. • Else R11 returns 0.		
Other	Unmodified		

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.VP.VEINFO.GET returns the virtualization exception information of a #VE exception that was previously delivered to the guest TD.

Enumeration: Support of version 1 is enumerated by TDX_FEATURES0.VE_REDUCTION (bit 30). Support of version 2 is enumerated by TDX_FEATURES0.APX (bit 28). TDX_FEATURES0 is readable by TDG.SYS.RD.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.380: TDG.VP.VEINFO.GET Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDVPS structure	TDVPS	RW	Opaque	N/A	Shared(i)

The function checks the following conditions (no specific order is implied):

- The VALID field in TDVPS.VE_INFO must non-0 to indicate that a valid virtualization information is available.

If successful, the function does the following:

1. Return the EXIT_REASON, EXIT_QUALIFICATION, GLA, GPA, INSTRUCTION_LENGTH and INSTRUCTION_INFORMATTION from TDVPS.VE_INFO in GPRs.
2. Clear the VALID field in TDVPS.VE_INFO to 0 to indicate that the virtualization information has been read.

Completion Status Codes

Table 5.381: TDG.VP.VEINFO.GET Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_NO_VE_INFO	There is no Virtualization Exception information.
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.VP.VEINFO.GET is successful.

5.5.21. TDG.VP.VMCALL Leaf

Perform a TD Exit to the host VMM.

Table 5.382: TDG.VP.VMCALL Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	A bitmap that controls which part of the guest TD GPR and XMM state is passed as-is to the VMM and back A bit value of 0 indicates that the corresponding register is saved by the Intel TDX module, scrubbed to 0 before SEAMRET to the host VMM, and restored by the Intel TDX module on the following TDH.VP.ENTER. A bit value of 1 indicates that the corresponding register is passed as-is to the host VMM, and on the following TDH.VP.ENTER, the register value is used as input from the host VMM and passed as-is to the guest TD. The value of RCX is passed to the host VMM.		
	Bits	**Name**	**Description**
	15:0	GPR Mask	Controls the transfer of GPR values: Bit 0: RAX – must be 0 Bit 1: RCX – must be 0 Bit 2: RDX Bit 3: RBX Bit 4: RSP – must be 0 Bit 5: RBP – if the TD's CONFIG_FLAG.NO_RBP_MOD is 1, then this bit must be 0. See the enumeration note below. Bit 6: RSI Bit 7: RDI Bits 15:8: R15 – R8
	31:16	XMM Mask	Controls the transfer of XMM15 – XMM0 register values
	63:32	Reserved	Reserved: must be 0
RBX, RDX, RBP, RSI, RDI, R8 – R15	If the corresponding bit in RCX is set to 1, the register value passed as-is to the host VMM on SEAMRET. Else, the register value is not used as an input and is preserved. If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP can't be used to pass values to the host VMM. See the enumeration note below.		
XMM0 – XMM15	If the corresponding bit in RCX is set to 1, the register value passed as-is to the host VMM on SEAMRET. Else, the register value is not used as an input and is preserved.		

Table 5.383: TDG.VP.VMCALL Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code: returns a constant value of TDX_SUCCESS (0)
RCX	Unmodified
RBX, RDX, RBP, RDI, RSI, R8 – R15	If the corresponding bit in RCX is set to 1, the register value passed as-is from the host VMM's SEAMCALL(TDH.VP.ENTER) input. Else, the register value is unmodified. If the TD's CONFIG_FLAGS.NO_RBP_MOD is set to 1, then RBP can't be used to pass values from the host VMM and is not modified from its input value. See the enumeration note below.
XMM0 – XMM15	If the corresponding bit in RCX is set to 1, the register value passed as-is from the host VMM's SEAMCALL(TDH.VP.ENTER) input. Else, the register value is unmodified.
Other	Unmodified

Leaf Function Description

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.VP.VMCALL performs a TD exit to the host VMM. From the VMM's point of view, this is the termination of a previous SEAMCALL(TDH.VP.ENTER). Selected GPR and XMM state is passed to the VMM host, controlled by RCX as shown above. The rest of the CPU state is saved in TDVPS and replaced with a synthetic state.

From the guest TD's point of view, a subsequent SEAMCALL(TDH.VP.ENTER) from the host VMM terminates the TDG.VP.VMCALL function. Most GPR state, and if the value of RCX bit 1 is set, all XMM state, is passed to the TD guest as shown above.

Enumeration: Control of RBP usage as an input/output parameter by the TD's CONFIG_FLAG.NO_RBP_MOD is enumerated by TDX_FEATURES0.NO_RBP_MOD (bit 18), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, then RBP can be used by TDG.VP.VMCALL to pass information between the guest TD and the host VMM, although highly discouraged since it contradicts normal calling conventions ABI.

L2 VM Details: TDG.VP.VMCALL may be invoked by an L2 VM, if enabled by the L1 VMM for the current VCPU (by setting TDVPS.L2_CTLS.ENABLE_TDVMCALL). If not enabled, then TDG.VP.VMCALL results in an L2→L1 exit.

On subsequent TD resumption, the host VMM may request resumption into L1 by setting TDH.VP.ENTER's RESUME_L1 flag. In this case, L1 is resumed (i.e., the TDG.VP.ENTER it has invoked is terminated) with a TDX_L2_EXIT_HOST_ROUTED_TDVMCALL status. The L2 VCPU state reflects the successful completion of TDG.VP.VMCALL.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.384: TDG.VP.VMCALL Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDVPS structure	TDVPS	RW	Opaque	N/A	Shared(i)

1. If invoked from an L2 VM, and TDVPS.L2_CTLS.ENABLE_TDVMCALL is 0, do an L2→L1 exit.
2. Save guest TD CPU state to TDVPS (including TD VMCS):
   2.1. Save extended state per TDCS.XFAM. There is no strict requirement to save XMM state that will be passed to the host VMM as controlled by RCX. This state will be overwritten on the next TD entry.
   2.2. Save GPR state. There is no strict requirement to save GPR state that will be passed to the host VMM as controlled by RCX (but RCX itself must be saved). This state will be overwritten on the next TD entry.
   2.3. Advance the saved RIP to the instruction following TDCALL.
3. Adjust the TDCS TLB tracking counters.
4. Release the shared locking – acquired on TDH.VP.ENTER of TDR, TDCS and TDVPS.
5. Load host VMM state:
   5.1. Clear the extended state except XMM (per TDCS.XFAM) to synthetic INIT values.
   5.2. As controlled by RCX, either clear or set to the guest TD's value the state of XMM0 – XMM15.
   5.3. As controlled by RCX, either clear or set to the guest TD's value the state of RBX, RDX, RBP, RDI, RSI and R8 – R15.
   5.4. Set RCX to the guest TD's value.
   5.5. Set RAX to the TDCALL exit reason.
   5.6. Restore other host VMM state – saved during TDH.VP.ENTER.
6. Execute SEAMRET to return to the host VMM.

Note: Logically, from the point of view of the guest TD, TDG.VP.VMCALL is terminated by the next TDH.VP.ENTER.

Completion Status Codes

Table 5.385: TDG.VP.VMCALL Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.VP.VMCALL is successful. TD exit was done, resulting a in a completion of SEAMCALL(TDH.VP.ENTER) on the host VMM side. Later, the host VMM executed SEAMCALL(TDH.VP.ENTER) again, and execution returned to the guest TD VCPU (in TDX non-root mode) completing TDG.VP.VMCALL.

5.5.22. TDG.VP.WR Leaf

Write a VCPU-scope metadata field (control structure field) of a TD.

Table 5.386: TDG.VP.WR Input Operands

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 5.5.1		
	Bits	**Field**	**Description**
	15:0	Leaf Number	Selects the TDCALL interface function
	23:16	Version Number	Selects the TDCALL interface function version Must be 0
	63:24	Reserved	Must be 0
RCX	Reserved, must be 0		
RDX	Field identifier – see 3.10 The LAST_ELEMENT_IN_FIELD and LAST_FIELD_IN_SEQUENCE components of the field identifier must be 0. WRITE_MASK_VALID, INC_SIZE, CONTEXT_CODE and FIELD_SIZE components of the field identifier are ignored.		
R8	Data to write to the field		
R9	A 64b write mask to indicate which bits of the value in R8 are to be written to the field		

Table 5.387: TDG.VP.WR Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 5.5.1
R8	Previous contents of the field In case of an error, as indicated by RAX, R8 returns 0.
Other	Unmodified

Leaf Function Description

Intel SDM, Vol.3, Appendix A VMX Capabilities Reporting Facility

Note: The description below is provided at a high level. Actual details, order of checks, returned status codes, etc. may vary.

TDG.VP.WR writes a VCPU-scope metadata field (control structure field) of a TD. The value (R8) is written as specified by the write mask (R9). Writing is subject to the field's internal write mask (per the TD's ATTRIBUTES.DEBUG bit).

Table 5.388: Metadata Field Write Rules

Write Mask Bit in R9	Internal Write Mask Bit	Value Bit in R8
0	N/A	Silently ignored
1	0	Must be the same as the current field's bit
1	1	Written to the current field's bit

Writing of specific fields is also subject to additional rules, e.g.:

- Writing of L2 VMCS fields is subject to the VMX capabilities reported by the applicable virtual values of IA32_VMX_* MSRs, as described in [Intel SDM, Vol.3, Appendix A].
- Guest CR0 and CR4 values are subject to the CR0/4 guest host mask and read shadow settings. For details, see the [TD Partitioning Spec].

Enumeration: Availability of TDG.VP.WR enumerated by TDX_FEATURES0.ENHANCED_METADATA (bit 3), readable by TDH.SYS.RD* (see 3.3.3.1). If not supported, calling TDG.VP.WR returns a TDX_OPERAND_INVALID(RAX) status.

To understand the table and text below, please refer to the [TDX Module Base Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 5.389 TDG.VM.WR Operands Information Definition

Explicit/ Implicit	Reg.	Ref Type	Resource	Resource Type	Access	Access Semantics	Align Check	Concurrency Restrictions
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDVPS structure	TDVPS	RW	Opaque	N/A	Shared(i)

If the memory operand checks, per the table above, pass:

1. Write the control structure field and return its old value, using the algorithm described in 5.3.2.2.

Completion Status Codes

Table 5.390: TDG.VP.WR Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_METADATA_FIELD_ID_INCORRECT	
TDX_METADATA_FIELD_NOT_READABLE	
TDX_METADATA_FIELD_NOT_WRITABLE	
TDX_METADATA_FIELD_VALUE_NOT_VALID	
TDX_METADATA_WR_MASK_NOT_VALID	
TDX_OPERAND_BUSY	Operation encountered a busy operand, indicated by the lower 32 bits of the status. In many cases, this can be resolved by retrying the operation.
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.VP.WR is successful.
TDX_TD_VMCS_FIELD_NOT_INITIALIZED	

Intel TDX Application Binary Interface (ABI) Reference