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

348551-001US

September 2021

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

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 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 **ABI Reference Specification**	[TDX Module ABI]	Detailed TDX module Application Binary Interface (ABI) reference specification, covering the entire TDX module architecture

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 Spec].

1.3. Notation

See the [TDX Module Spec].

1.4. References

See the [TDX Module Spec].

2. ABI Reference: CPU Virtualization Tables

2.1. MSR Virtualization

Table 2.2 below describes how the Intel TDX module virtualizes MSRs to guest TDs. The table uses a notation that is described in Table 2.1 below.

Table 2.1: MSR Virtualization Notation Definition

Text	Virtualization
Native	Direct read or write from/to CPU

Inject_GP(condition)	TDX Module injects a #GP(0) if condition is true, else reads from CPU or write to CPU:
	`if (condition)`
	`  #GP(0)`
	`else`
	`  Native`

Inject_GP_or_VE(condition)	TDX Module injects a #GP(0) if condition is true, else it injects a #VE:
	`if (condition)`
	`  #GP(0)`
	`else`
	`  #VE`

For MSRs that are not listed in the table, the Intel TDX module injects a #VE on both RDMSR and WRMSR by the guest TD.

Note: The table below provides a high-level overview of MSR virtualization. Implementation details may differ.

Table 2.2: MSR Virtualization

MSR Index Range (Hex)				MSR Virtualization	
First (Hex)	**Last (Hex)**	**Size (Hex)**	**MSR Architectural Name**	**On RDMSR**	**On WRMSR**
0x0010	0x0010	0x1	IA32_TIME_STAMP_COUNTER	Native	#VE
0x0048	0x0048	0x1	IA32_SPEC_CTRL	Native	Native
0x0049	0x0049	0x1	IA32_PRED_CMD	Native	Native
0x0087	0x0087	0x1	IA32_MKTME_PARTITIONING	Inject_GP_or_VE (~virt. CPUID(7,0).EDX[18])	Inject_GP_or_VE (~virt. CPUID(7,0).EDX[18])
0x008C	0x008F	0x4	IA32_SGXLEPUBKEYHASHx	#GP(0)	#GP(0)
0x0098	0x0098	0x1	MSR_WBINVDP	#GP(0)	#GP(0)
0x0099	0x0099	0x1	MSR_WBNOINVDP	#GP(0)	#GP(0)
0x009A	0x009A	0x1	MSR_INTR_PENDING	#GP(0)	#GP(0)
0x009B	0x009B	0x1	IA32_SMM_MONITOR_CTL	#GP(0)	#GP(0)
0x009E	0x009E	0x1	IA32_SMBASE	#GP(0)	#GP(0)
0x00C1	0x00C8	0x8	IA32_PMCx	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)
0x00E1	0x00E1	0x1	IA32_UMWAIT_CONTROL	Inject_GP(~virt. CPUID(7,0).ECX[5])	Inject_GP(~virt. CPUID(7,0).ECX[5])
0x010A	0x010A	0x1	IA32_ARCH_CAPABILITIES	Get the value read on TDX module init Set bit 7 (TSX_CTRL) = 0	Native
0x010B	0x010B	0x1	IA32_FLUSH_CMD	Native	Native
0x0122	0x0122	0x1	IA32_TSX_CTRL	#GP(0)	#GP(0)
0x0174	0x0174	0x1	IA32_SYSENTER_CS	Native	Native
0x0175	0x0175	0x1	IA32_SYSENTER_ESP	Native	Native
0x0176	0x0176	0x1	IA32_SYSENTER_EIP	Native	Native
0x0186	0x018D	0x8	IA32_PERFEVTSELx	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)
0x01A0	0x01A0	0x1	IA32_MISC_ENABLE	if ~PERFMON RDMSR current value Indicate Perfmon and PEBS are unavailable: Bit 7 = 0 Bit 12 = 1 else Native	#VE
0x01A6	0x01A7	0x2	MSR_OFFCORE_RSPx	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)

MSR Index Range (Hex)				MSR Virtualization	
First (Hex)	Last (Hex)	Size (Hex)	MSR Architectural Name	On RDMSR	On WRMSR
0x01C4	0x01C4	0x1	IA32_XFD	Inject_GP(~(virt. CPUID(0xD,0x1).EAX[4]))	Inject_GP(~(virt. CPUID(0xD,0x1).EAX[4]))
0x01C5	0x01C5	0x1	IA32_XFD_ERR	Inject_GP(~(virt. CPUID(0xD,0x1).EAX[4]))	Inject_GP(~(virt. CPUID(0xD,0x1).EAX[4]))
0x01D9	0x01D9	0x1	IA32_DEBUGCTL	Clear ENABLE_UNCORE_PMI (bit 13)	#GP if illegal, #VE if value is not supported for TD
0x01F8	0x01F8	0x1	IA32_PLATFORM_DCA_CAP	Inject_GP_or_VE(~virt. CPUID(0x1).ECX[18])	Inject_GP_or_VE(~virt. CPUID(0x1).ECX[18])
0x01F9	0x01F9	0x1	IA32_CPU_DCA_CAP	Inject_GP_or_VE(~virt. CPUID(0x1).ECX[18])	Inject_GP_or_VE(~virt. CPUID(0x1).ECX[18])
0x01FA	0x01FA	0x1	IA32_DCA_0_CAP	Inject_GP_or_VE(~virt. CPUID(0x1).ECX[18])	Inject_GP_or_VE(~virt. CPUID(0x1).ECX[18])
0x0276	0x0276	0x1	MSR_SLAM_ENABLE	#GP(0)	#GP(0)
0x0277	0x0277	0x1	IA32_PAT	Native	Native
0x0309	0x030C	0x4	IA32_FIXED_CTRx	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)
0x0329	0x0329	0x1	IA32_PERF_METRICS	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)

0x0345	0x0345	0x1	IA32_PERF_CAPABILITIES	if ~PERFMON
				  return 0
				else if ~XFAM[8]
				  clear bit 16
				else
				  Native
					Inject_GP(~PERFMON)

0x038D	0x038D	0x1	IA32_FIXED_CTR_CTRL	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)
0x038E	0x038E	0x1	IA32_PERF_GLOBAL_STATUS	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)
0x038F	0x038F	0x1	IA32_PERF_GLOBAL_CTRL	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)
0x0390	0x0390	0x1	IA32_PERF_GLOBAL_STATUS_RESET	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)
0x0391	0x0391	0x1	IA32_PERF_GLOBAL_STATUS_SET	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)
0x0392	0x0392	0x1	IA32_PERF_GLOBAL_INUSE	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)
0x03F1	0x03F1	0x1	IA32_PEBS_ENABLE	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)
0x03F2	0x03F2	0x1	MSR_PEBS_DATA_CFG	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)
0x03F6	0x03F6	0x1	MSR_PEBS_LD_LAT	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)
0x03F7	0x03F7	0x1	MSR_PEBS_FRONTEND	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)
0x0480	0x0480	0x1	IA32_VMX_BASIC	#GP(0)	#GP(0)
0x0481	0x0481	0x1	IA32_VMX_PINBASED_CTLS	#GP(0)	#GP(0)
0x0482	0x0482	0x1	IA32_VMX_PROCBASED_CTLS	#GP(0)	#GP(0)
0x0483	0x0483	0x1	IA32_VMX_EXIT_CTLS	#GP(0)	#GP(0)
0x0484	0x0484	0x1	IA32_VMX_ENTRY_CTLS	#GP(0)	#GP(0)
0x0485	0x0485	0x1	IA32_VMX_MISC	#GP(0)	#GP(0)
0x0486	0x0486	0x1	IA32_VMX_CR0_FIXED0	#GP(0)	#GP(0)
0x0487	0x0487	0x1	IA32_VMX_CR0_FIXED1	#GP(0)	#GP(0)
0x0488	0x0488	0x1	IA32_VMX_CR4_FIXED0	#GP(0)	#GP(0)
0x0489	0x0489	0x1	IA32_VMX_CR4_FIXED1	#GP(0)	#GP(0)
0x048A	0x048A	0x1	IA32_VMX_VMCS_ENUM	#GP(0)	#GP(0)
0x048B	0x048B	0x1	IA32_VMX_PROCBASED_CTLS2	#GP(0)	#GP(0)
0x048C	0x048C	0x1	IA32_VMX_EPT_VPID_CAP	#GP(0)	#GP(0)
0x048D	0x048D	0x1	IA32_VMX_TRUE_PINBASED_CTLS	#GP(0)	#GP(0)
0x048E	0x048E	0x1	IA32_VMX_TRUE_PROCBASED_CTLS	#GP(0)	#GP(0)
0x048F	0x048F	0x1	IA32_VMX_TRUE_EXIT_CTLS	#GP(0)	#GP(0)
0x0490	0x0490	0x1	IA32_VMX_TRUE_ENTRY_CTLS	#GP(0)	#GP(0)
0x0491	0x0491	0x1	IA32_VMX_VMFUNC	#GP(0)	#GP(0)
0x0492	0x0492	0x1	IA32_VMX_PROCBASED_CTLS3	#GP(0)	#GP(0)
0x04C1	0x04C8	0x8	IA32_A_PMCx	Inject_GP(~PERFMON)	Inject_GP(~PERFMON)
0x0500	0x0500	0x1	IA32_SGX_SVN_STATUS	#GP(0)	#GP(0)
0x0560	0x0560	0x1	IA32_RTIT_OUTPUT_BASE	Inject_GP(~XFAM[8])	Inject_GP(~XFAM[8])
0x0561	0x0561	0x1	IA32_RTIT_OUTPUT_MASK_PTRS	Inject_GP(~XFAM[8])	Inject_GP(~XFAM[8])
0x0570	0x0570	0x1	IA32_RTIT_CTL	Inject_GP(~XFAM[8])	Inject_GP(~XFAM[8])
0x0571	0x0571	0x1	IA32_RTIT_STATUS	Inject_GP(~XFAM[8])	Inject_GP(~XFAM[8])
0x0572	0x0572	0x1	IA32_RTIT_CR3_MATCH	Inject_GP(~XFAM[8])	Inject_GP(~XFAM[8])
0x0580	0x0580	0x1	IA32_RTIT_ADDR0_A	Inject_GP(~XFAM[8])	Inject_GP(~XFAM[8])
0x0581	0x0581	0x1	IA32_RTIT_ADDR0_B	Inject_GP(~XFAM[8])	Inject_GP(~XFAM[8])
0x0582	0x0582	0x1	IA32_RTIT_ADDR1_A	Inject_GP(~XFAM[8])	Inject_GP(~XFAM[8])
0x0583	0x0583	0x1	IA32_RTIT_ADDR1_B	Inject_GP(~XFAM[8])	Inject_GP(~XFAM[8])
0x0584	0x0584	0x1	IA32_RTIT_ADDR2_A	Inject_GP(~XFAM[8])	Inject_GP(~XFAM[8])
0x0585	0x0585	0x1	IA32_RTIT_ADDR2_B	Inject_GP(~XFAM[8])	Inject_GP(~XFAM[8])
0x0586	0x0586	0x1	IA32_RTIT_ADDR3_A	Inject_GP(~XFAM[8])	Inject_GP(~XFAM[8])
0x0587	0x0587	0x1	IA32_RTIT_ADDR3_B	Inject_GP(~XFAM[8])	Inject_GP(~XFAM[8])
0x0600	0x0600	0x1	IA32_DS_AREA	Native	Native

MSR Index Range (Hex)				MSR Virtualization	
First (Hex)	Last (Hex)	Size (Hex)	MSR Architectural Name	On RDMSR	On WRMSR
0x06A0	0x06A0	0x1	IA32_U_CET	Inject_GP(~(XFAM[11] | XFAM[12]))	Inject_GP(~(XFAM[11] | XFAM[12]))
0x06A2	0x06A2	0x1	IA32_S_CET	Inject_GP(~(XFAM[11] | XFAM[12]))	Inject_GP(~(XFAM[11] | XFAM[12]))
0x06A4	0x06A4	0x1	IA32_PL0_SSP	Inject_GP(~(XFAM[11] | XFAM[12]))	Inject_GP(~(XFAM[11] | XFAM[12]))
0x06A5	0x06A5	0x1	IA32_PL1_SSP	Inject_GP(~(XFAM[11] | XFAM[12]))	Inject_GP(~(XFAM[11] | XFAM[12]))
0x06A6	0x06A6	0x1	IA32_PL2_SSP	Inject_GP(~(XFAM[11] | XFAM[12]))	Inject_GP(~(XFAM[11] | XFAM[12]))
0x06A7	0x06A7	0x1	IA32_PL3_SSP	Inject_GP(~(XFAM[11] | XFAM[12]))	Inject_GP(~(XFAM[11] | XFAM[12]))
0x06A8	0x06A8	0x1	IA32_INTERRUPT_SSP_TABLE_ADDR	Inject_GP(~(XFAM[11] | XFAM[12]))	Inject_GP(~(XFAM[11] | XFAM[12]))
0x06E1	0x06E1	0x1	IA32_PKRS	Inject_GP(~PKS)	Inject_GP(~PKS)
0x0800	0x0801	0x2	Reserved for xAPIC MSRs	#GP(0)	#GP(0)
0x0804	0x0807	0x4	Reserved for xAPIC MSRs	#GP(0)	#GP(0)
0x0808	0x0808	0x1	IA32_X2APIC_TPR	Native	Native
0x0809	0x0809	0x1	Reserved for xAPIC MSRs	Native	Native
0x080A	0x080A	0x1	IA32_X2APIC_PPR	Native	Native
0x080B	0x080B	0x1	IA32_X2APIC_EOI	Native	Native
0x080C	0x080C	0x1	Reserved for xAPIC MSRs	Native	Native
0x080E	0x080E	0x1	Reserved for xAPIC MSRs	Native	Native
0x0810	0x0817	0x8	IA32_X2APIC_ISRx	Native	Native
0x0818	0x081F	0x8	IA32_X2APIC_TMRx	Native	Native
0x0820	0x0827	0x8	IA32_X2APIC_IRRx	Native	Native
0x0829	0x082E	0x6	Reserved for xAPIC MSRs	#GP(0)	#GP(0)
0x0831	0x0831	0x1	Reserved for xAPIC MSRs	#GP(0)	#GP(0)
0x083F	0x083F	0x1	IA32_X2APIC_SELF_IPI	Native	Native
0x0840	0x087F	0x40	Reserved for xAPIC MSRs	#GP(0)	#GP(0)
0x0880	0x08BF	0x40	Reserved for xAPIC MSRs	#GP(0)	#GP(0)
0x08C0	0x08FF	0x40	Reserved for xAPIC MSRs	#GP(0)	#GP(0)
0x0981	0x0981	0x1	IA32_TME_CAPABILITY	Inject_GP_or_VE (~virt. CPUID(7,0).ECX[13])	Inject_GP_or_VE (~virt. CPUID(7,0).ECX[13])
0x0982	0x0982	0x1	IA32_TME_ACTIVATE	Inject_GP_or_VE (~virt. CPUID(7,0).ECX[13])	Inject_GP_or_VE (~virt. CPUID(7,0).ECX[13])
0x0983	0x0983	0x1	IA32_TME_EXCLUDE_MASK	Inject_GP_or_VE (~virt. CPUID(7,0).ECX[13])	Inject_GP_or_VE (~virt. CPUID(7,0).ECX[13])
0x0984	0x0984	0x1	IA32_TME_EXCLUDE_BASE	Inject_GP_or_VE (~virt. CPUID(7,0).ECX[13])	Inject_GP_or_VE (~virt. CPUID(7,0).ECX[13])
0x0985	0x0985	0x1	IA32_UINT_RR	Inject_GP(~XFAM[14])	Inject_GP(~XFAM[14])
0x0986	0x0986	0x1	IA32_UINT_HANDLER	Inject_GP(~XFAM[14])	Inject_GP(~XFAM[14])
0x0987	0x0987	0x1	IA32_UINT_STACKADJUST	Inject_GP(~XFAM[14])	Inject_GP(~XFAM[14])
0x0988	0x0988	0x1	IA32_UINT_MISC	Inject_GP(~XFAM[14])	Inject_GP(~XFAM[14])
0x0989	0x0989	0x1	IA32_UINT_PD	Inject_GP(~XFAM[14])	Inject_GP(~XFAM[14])
0x098A	0x098A	0x1	IA32_UINT_TT	Inject_GP(~XFAM[14])	Inject_GP(~XFAM[14])
0x0C80	0x0C80	0x1	IA32_DEBUG_INTERFACE	Native	#VE
0x0D90	0x0D90	0x1	IA32_BNDCFGS	#GP(0)	#GP(0)
0x0D93	0x0D93	0x1	IA32_PASID	#GP(0)	#GP(0)

0x0DA0	0x0DA0	0x1	IA32_XSS	Native	if illegal or does not match XFAM
					  #GP(0)
					else
					  Write to CPU

0x1200	0x12FF	0x100	IA32_LBR_INFO	Inject_GP(~XFAM[15])	Inject_GP(~XFAM[15])
0x14CE	0x14CE	0x1	IA32_LBR_CTL	Inject_GP(~XFAM[15])	Inject_GP(~XFAM[15])
0x14CF	0x14CF	0x1	IA32_LBR_DEPTH	Inject_GP(~XFAM[15])	Inject_GP(~XFAM[15])
0x1500	0x15FF	0x100	IA32_LBR_FROM_IP	Inject_GP(~XFAM[15])	Inject_GP(~XFAM[15])
0x1600	0x16FF	0x100	IA32_LBR_TO_IP	Inject_GP(~XFAM[15])	Inject_GP(~XFAM[15])
0xC0000080	0xC0000080	0x1	IA32_EFER	Native	#VE
0xC0000081	0xC0000081	0x1	IA32_STAR	Native	Native
0xC0000082	0xC0000082	0x1	IA32_LSTAR	Native	Native
0xC0000084	0xC0000084	0x1	IA32_FMASK	Native	Native
0xC0000100	0xC0000100	0x1	IA32_FSBASE	Native	Native
0xC0000101	0xC0000101	0x1	IA32_GSBASE	Native	Native
0xC0000102	0xC0000102	0x1	IA32_KERNEL_GS_BASE	Native	Native
0xC0000103	0xC0000103	0x1	IA32_TSC_AUX	Native	Native

2.2. UPDATED: CPUID Virtualization

Table 2.4 below describes how the Intel TDX module virtualizes CPUID to guest TDs. Note the following:

- The "Configuration by TDH.MNG.INIT" column details which section of the TD_PARAMS structure is used for configuring how each CPUID bit field is virtualized.
- The "Virtualization" column uses a notation defined in Table 2.3 below.
- If the guest TD executes CPUID with a valid leaf / sub-leaf number combination that is not listed in the table, the Intel TDX module injects a #VE.
- The host VMM should always consult the list of CPUID leaves and sub-leaves configured by TD_PARAMS.CPUID_CONFIG, as enumerated by TDH.SYS.RD/RDALL or TDH.SYS.INFO.

Table 2.3: CPUID Virtualization Notation Definition

CPUID Bit Field Virtualization	Meaning	Virtualization Details
As Configured	Virtual bit field value reflects the host VMM configuration.	
As Configured (if Native)	If the native bit field value returned by executing CPUID is 0, then the virtual bit field value is o. Else, the virtual bit field value reflects the host VMM configuration.	
Calculated	Bit field is calculated by the Intel TDX module.	Calculation method
Fixed	The virtual bit field value is fixed.	Bit field value
Native	The virtual bit field value reflects the native value returned by executing CPUID.	

Note: The table below provides a high-level overview of CPUID virtualization. Implementation details may differ.

Table 2.4: CPUID Virtualization Overview

CPUID Field					Configuration by TDH.MNG.INIT		Virtualization	
Reg.	MSB	LSB	Field Size	Field Name	TD_PARAMS Section(s)	Configuration Details	Virtualization Type	Virtualization Details
Leaf 0x0								
EAX	31	0	32	MaxIndex	N/A		Fixed	0x21
EBX	31	0	32	Genu	N/A		Native	
ECX	31	0	32	ntel	N/A		Native	
EDX	31	0	32	ineI	N/A		Native	
Leaf 0x1								
EAX	3	0	4	Stepping ID	N/A		Calculated	Min. of all packages
EAX	7	4	4	Model ID	N/A		Native	
EAX	11	8	4	Family ID	N/A		Native	
EAX	13	12	2	Processor Type	N/A		Native	
EAX	15	14	2	Reserved	N/A		Fixed	0x0
EAX	19	16	4	Extended Model ID	N/A		Native	
EAX	27	20	8	Extended Family ID	N/A		Native	
EAX	31	28	4	Reserved	N/A		Fixed	0x0
EBX	7	0	8	Brand Index	N/A		Native	
EBX	15	8	8	CLFLUSH Line Size	N/A		Fixed	0x8
EBX	23	16	8	Maximum Addressable IDs	CPUID_CONFIG		As Configured	
EBX	31	24	8	Initial APIC ID	N/A		Calculated	TDVPS.VCPU_INDEX[7:0]
ECX	0	0	1	SSE3	N/A		Native	
ECX	1	1	1	PCLMULQDQ	N/A		Native	
ECX	2	2	1	DTES64	N/A		Native	
ECX	3	3	1	MONITOR	N/A		Fixed	0x0
ECX	4	4	1	DS-CPL	N/A		Native	
ECX	5	5	1	VMX	N/A		Fixed	0x0
ECX	6	6	1	SMX	N/A		Fixed	0x0
ECX	7	7	1	EST	CPUID_CONFIG		As Configured (if Native)	
ECX	8	8	1	TM2	CPUID_CONFIG		As Configured (if Native)	
ECX	9	9	1	SSSE3	N/A		Native	
ECX	10	10	1	CNXT-ID	N/A		Native	
ECX	11	11	1	SDBG	N/A		Native	
ECX	12	12	1	FMA	XFAM	XFAM[2]	As Configured (if Native)	
ECX	13	13	1	CMPXCHG16B	N/A		Fixed	0x1

CPUID Field					Configuration by TDH.MNG.INIT		Virtualization	
Reg.	MSB	LSB	Field Size	Field Name	TD_PARAMS Section(s)	Configuration Details	Virtualization Type	Virtualization Details
ECX	14	14	1	xTPR Update Control	CPUID_CONFIG		As Configured (if Native)	
ECX	15	15	1	PDCM	N/A		Fixed	0x1
ECX	16	16	1	Reserved	N/A		Fixed	0x0
ECX	17	17	1	PCID	N/A		Native	
ECX	18	18	1	DCA	CPUID_CONFIG		As Configured (if Native)	
ECX	19	19	1	SSE4_1	N/A		Native	
ECX	20	20	1	SSE4_2	N/A		Native	
ECX	21	21	1	x2APIC	N/A		Fixed	0x1
ECX	22	22	1	MOVBE	N/A		Native	
ECX	23	23	1	POPCNT	N/A		Native	
ECX	24	24	1	TSC-Deadline	N/A		Native	
ECX	25	25	1	AESNI	N/A		Fixed	0x1
ECX	26	26	1	XSAVE	N/A		Fixed	0x1
ECX	27	27	1	OSXSAVE	N/A		Calculated	CR4.OSXSAVE
ECX	28	28	1	AVX	XFAM, CPUID_CONFIG	XFAM[2]	As Configured (if Native)	
ECX	29	29	1	F16C	XFAM, CPUID_CONFIG	XFAM[2]	As Configured (if Native)	
ECX	30	30	1	RDRAND	N/A		Fixed	0x1
ECX	31	31	1	Reserved	N/A		Fixed	0x0
EDX	0	0	1	FPU	N/A		Native	
EDX	1	1	1	VME	N/A		Native	
EDX	2	2	1	DE	N/A		Native	
EDX	3	3	1	PSE	N/A		Native	
EDX	4	4	1	TSC	N/A		Native	
EDX	5	5	1	MSR	N/A		Fixed	0x1
EDX	6	6	1	PAE	N/A		Fixed	0x1
EDX	7	7	1	MCE	N/A		Fixed	0x1
EDX	8	8	1	CX8	N/A		Native	
EDX	9	9	1	APIC	N/A		Fixed	0x1
EDX	10	10	1	Reserved	N/A		Fixed	0x0
EDX	11	11	1	SEP	N/A		Native	
EDX	12	12	1	MTRR	N/A		Fixed	0x1
EDX	13	13	1	PGE	N/A		Native	
EDX	14	14	1	MCA	N/A		Fixed	0x1
EDX	15	15	1	CMOV	N/A		Native	
EDX	16	16	1	PAT	N/A		Native	
EDX	17	17	1	PSE-36	N/A		Native	
EDX	18	18	1	PSN	N/A		Native	
EDX	19	19	1	CLFSH	N/A		Fixed	0x1
EDX	20	20	1	Reserved	N/A		Fixed	0x0
EDX	21	21	1	DS	N/A		Fixed	0x1
EDX	22	22	1	ACPI	CPUID_CONFIG		As Configured (if Native)	
EDX	23	23	1	MMX	N/A		Native	
EDX	24	24	1	FXSR	N/A		Native	
EDX	25	25	1	SSE	N/A		Native	
EDX	26	26	1	SSE2	N/A		Native	
EDX	27	27	1	SS	N/A		Native	
EDX	28	28	1	HTT	CPUID_CONFIG		As Configured (if Native)	
EDX	29	29	1	TM	CPUID_CONFIG		As Configured (if Native)	
EDX	30	30	1	Reserved	N/A		Fixed	0x0
EDX	31	31	1	PBE	CPUID_CONFIG		As Configured (if Native)	
				Leaf 0x3				
EAX	31	0	32	Reserved	N/A		Fixed	0x0
EBX	31	0	32	Reserved	N/A		Fixed	0x0
ECX	31	0	32	Reserved	N/A		Fixed	0x0
EDX	31	0	32	Reserved	N/A		Fixed	0x0
				Leaf 0x4 / Sub-Leaf 0x0				
EAX	4	0	5	Type	CPUID_CONFIG		As Configured	
EAX	7	5	3	Level	CPUID_CONFIG		As Configured	
EAX	8	8	1	Self Initializing	CPUID_CONFIG		As Configured	
EAX	9	9	1	Fully Associative	CPUID_CONFIG		As Configured	
EAX	13	10	4	Reserved	N/A		Fixed	0x0
EAX	25	14	12	Addressable IDs Sharing this Cache	CPUID_CONFIG		As Configured	
EAX	31	26	6	Addressable IDs for Cores in Package	CPUID_CONFIG		As Configured	
EBX	11	0	12	L	N/A		Native	
EBX	21	12	10	P	CPUID_CONFIG		As Configured	
EBX	31	22	10	W	CPUID_CONFIG		As Configured	

CPUID Field					Configuration by TDH.MNG.INIT		Virtualization	
Reg.	MSB	LSB	Field Size	Field Name	TD_PARAMS Section(s)	Configuration Details	Virtualization Type	Virtualization Details
ECX	31	0	32	Number of Sets	CPUID_CONFIG		As Configured	
EDX	0	0	1	WBINVD	CPUID_CONFIG		As Configured	
EDX	1	1	1	Cache Inclusiveness	CPUID_CONFIG		As Configured	
EDX	2	2	1	Reserved	N/A		Fixed	0x0
EDX	31	3	29		CPUID_CONFIG		As Configured	
				Leaf 0x4 / Sub-Leaf 0x1				
EAX	4	0	5	Type	CPUID_CONFIG		As Configured	
EAX	7	5	3	Level	CPUID_CONFIG		As Configured	
EAX	8	8	1	Self Initializing	CPUID_CONFIG		As Configured	
EAX	9	9	1	Fully Associative	CPUID_CONFIG		As Configured	
EAX	13	10	4	Reserved	N/A		Fixed	0x0
EAX	25	14	12	Addressable IDs Sharing this Cache	CPUID_CONFIG		As Configured	
EAX	31	26	6	Addressable IDs for Cores in Package	CPUID_CONFIG		As Configured	
EBX	11	0	12	L	N/A		Native	
EBX	21	12	10	P	CPUID_CONFIG		As Configured	
EBX	31	22	10	W	CPUID_CONFIG		As Configured	
ECX	31	0	32	Number of Sets	CPUID_CONFIG		As Configured	
EDX	0	0	1	WBINVD	CPUID_CONFIG		As Configured	
EDX	1	1	1	Cache Inclusiveness	CPUID_CONFIG		As Configured	
EDX	2	2	1	Reserved	N/A		Fixed	0x0
EDX	31	3	29		CPUID_CONFIG		As Configured	
				Leaf 0x4 / Sub-Leaf 0x2				
EAX	4	0	5	Type	CPUID_CONFIG		As Configured	
EAX	7	5	3	Level	CPUID_CONFIG		As Configured	
EAX	8	8	1	Self Initializing	CPUID_CONFIG		As Configured	
EAX	9	9	1	Fully Associative	CPUID_CONFIG		As Configured	
EAX	13	10	4	Reserved	N/A		Fixed	0x0
EAX	25	14	12	Addressable IDs Sharing this Cache	CPUID_CONFIG		As Configured	
EAX	31	26	6	Addressable IDs for Cores in Package	CPUID_CONFIG		As Configured	
EBX	11	0	12	L	N/A		Native	
EBX	21	12	10	P	CPUID_CONFIG		As Configured	
EBX	31	22	10	W	CPUID_CONFIG		As Configured	
ECX	31	0	32	Number of Sets	CPUID_CONFIG		As Configured	
EDX	0	0	1	WBINVD	CPUID_CONFIG		As Configured	
EDX	1	1	1	Cache Inclusiveness	CPUID_CONFIG		As Configured	
EDX	2	2	1	Reserved	N/A		Fixed	0x0
EDX	31	3	29		CPUID_CONFIG		As Configured	
				Leaf 0x4 / Sub-Leaf 0x3				
EAX	4	0	5	Type	CPUID_CONFIG		As Configured	
EAX	7	5	3	Level	CPUID_CONFIG		As Configured	
EAX	8	8	1	Self Initializing	CPUID_CONFIG		As Configured	
EAX	9	9	1	Fully Associative	CPUID_CONFIG		As Configured	
EAX	13	10	4	Reserved	N/A		Fixed	0x0
EAX	25	14	12	Addressable IDs Sharing this Cache	CPUID_CONFIG		As Configured	
EAX	31	26	6	Addressable IDs for Cores in Package	CPUID_CONFIG		As Configured	
EBX	11	0	12	L	N/A		Native	
EBX	21	12	10	P	CPUID_CONFIG		As Configured	
EBX	31	22	10	W	CPUID_CONFIG		As Configured	
ECX	31	0	32	Number of Sets	CPUID_CONFIG		As Configured	
EDX	0	0	1	WBINVD	CPUID_CONFIG		As Configured	
EDX	1	1	1	Cache Inclusiveness	CPUID_CONFIG		As Configured	
EDX	2	2	1	Complex cache indexing	CPUID_CONFIG		As Configured	
EDX	31	3	29	Reserved	N/A		Fixed	0x0
				Leaf 0x4 / Sub-Leaf 0x4				
EAX	4	0	5	Type	N/A		Fixed	0x0
EAX	7	5	3	Level	N/A		Fixed	0x0
EAX	8	8	1	Self Initializing	N/A		Fixed	0x0
EAX	9	9	1	Fully Associative	N/A		Fixed	0x0
EAX	13	10	4	Reserved	N/A		Fixed	0x0
EAX	25	14	12	Addressable IDs Sharing this Cache	N/A		Fixed	0x0
EAX	31	26	6	Addressable IDs for Cores in Package	N/A		Fixed	0x0
EBX	11	0	12	L	N/A		Fixed	0x0
EBX	21	12	10	P	N/A		Fixed	0x0
EBX	31	22	10	W	N/A		Fixed	0x0
ECX	31	0	32	Number of Sets	N/A		Fixed	0x0
EDX	0	0	1	WBINVD	N/A		Fixed	0x0
EDX	1	1	1	Cache Inclusiveness	N/A		Fixed	0x0
EDX	2	2	1	Complex Cache Indexing	N/A		Fixed	0x0

CPUID Field					Configuration by TDH.MNG.INIT		Virtualization	
Reg.	MSB	LSB	Field Size	Field Name	TD_PARAMS Section(s)	Configuration Details	Virtualization Type	Virtualization Details
EDX	31	3	29	Reserved	N/A		Fixed	0x0
Leaf 0x7 / Sub-Leaf 0x0								
EAX	31	0	32	Max Sub-Leaves	N/A		Fixed	0x1
EBX	0	0	1	FSGSBASE	N/A		Fixed	0x1
EBX	1	1	1	IA32_TSC_ADJUST	N/A		Fixed	0x0
EBX	2	2	1	SGX	N/A		Fixed	0x0
EBX	3	3	1	BMI1	CPUID_CONFIG		As Configured (if Native)	
EBX	4	4	1	HLE	N/A		Native	
EBX	5	5	1	AVX2	XFAM	XFAM[2]	As Configured (if Native)	
EBX	6	6	1	FDP_EXCPTN_ONLY	N/A		Native	
EBX	7	7	1	SMEP	N/A		Native	
EBX	8	8	1	BMI2	CPUID_CONFIG		As Configured (if Native)	
EBX	9	9	1	Enhanced REP MOVSB/STOSB	N/A		Native	
EBX	10	10	1	INVPCID	N/A		Native	
EBX	11	11	1	RTM	N/A		Fixed	0x1
EBX	12	12	1	PQM	CPUID_CONFIG		As Configured (if Native)	
EBX	13	13	1	FCS/FDS Deprecation	N/A		Native	
EBX	14	14	1	MPX	N/A		Fixed	0x0
EBX	15	15	1	Cache QoS Enforcement	CPUID_CONFIG		As Configured (if Native)	
EBX	16	16	1	AVX512F	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
EBX	17	17	1	AVX512DQ	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
EBX	18	18	1	RDSEED	N/A		Fixed	0x1
EBX	19	19	1	ADCX/ADOX	CPUID_CONFIG		As Configured (if Native)	
EBX	20	20	1	SMAP/CLAC/STAC	N/A		Fixed	0x1
EBX	21	21	1	AVX512_IFMA	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
EBX	22	22	1	PCOMMIT	N/A		Native	
EBX	23	23	1	CLFLUSHOPT	N/A		Fixed	0x1
EBX	24	24	1	CLWB	N/A		Fixed	0x1
EBX	25	25	1	RTIT	XFAM	XFAM[8]	As Configured (if Native)	
EBX	26	26	1	AVX512PF	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
EBX	27	27	1	AVX512ER	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
EBX	28	28	1	AVX512CD	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
EBX	29	29	1	SHA	N/A		Fixed	0x1
EBX	30	30	1	AVX512BW	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
EBX	31	31	1	AVX512VL	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
ECX	0	0	1	PREFETCHWT1	N/A		Native	
ECX	1	1	1	AVX512VBMI	XFAM	XFAM[7:5]	As Configured (if Native)	
ECX	2	2	1	UMIP	N/A		Native	
ECX	3	3	1	PKU	XFAM	XFAM[9]	As Configured (if Native)	
ECX	4	4	1	OSPKE	N/A		Calculated	CR4.PKE
ECX	5	5	1	MONITORX/MWAITX	CPUID_CONFIG		As Configured (if Native)	
ECX	6	6	1	AVX512_VBMI2	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
ECX	7	7	1	CET Shadow Stack	XFAM	XFAM[12:11]	As Configured (if Native)	
ECX	8	8	1	GFNI	N/A		Native	
ECX	9	9	1	VAES	XFAM, CPUID_CONFIG	XFAM[2]	As Configured (if Native)	
ECX	10	10	1	VPCLMULQDQ	XFAM, CPUID_CONFIG	XFAM[2]	As Configured (if Native)	
ECX	11	11	1	AVX512_VNNI	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
ECX	12	12	1	AVX512_BITALG	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
ECX	13	13	1	TME	CPUID_CONFIG		As Configured (if Native)	
ECX	14	14	1	AVX512_VPOPCNTDQ	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
ECX	15	15	1	FZM	N/A		Fixed	0x0
ECX	16	16	1	57 bit Address Support	N/A		Native	
ECX	21	17	5	MAWAU for MPX	N/A		Fixed	0x0
ECX	22	22	1	RDPID	N/A		Native	
ECX	23	23	1	KL_ENABLED	ATTRIBUTES	KL	As Configured (if Native)	

CPUID Field					Configuration by TDH.MNG.INIT		Virtualization	
Reg.	MSB	LSB	Field Size	Field Name	TD_PARAMS Section(s)	Configuration Details	Virtualization Type	Virtualization Details
ECX	24	24	1	BUSLOCK	N/A		Fixed	0x1
ECX	25	25	1	CLDEMOTE	N/A		Native	
ECX	26	26	1	Reserved	N/A		Fixed	0x0
ECX	27	27	1	MOVDIRI	N/A		Native	
ECX	28	28	1	MOVDIR64B	N/A		Fixed	0x1
ECX	29	29	1	ENQCMD	N/A		Fixed	0x0
ECX	30	30	1	SGX_LC	N/A		Fixed	0x0
ECX	31	31	1	PKS	ATTRIBUTES	PKS	As Configured (if Native)	
EDX	0	0	1	Reserved	N/A		Fixed	0x0
EDX	1	1	1	Reserved	N/A		Fixed	0x0
EDX	2	2	1	AVX512_4VNNIW	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
EDX	3	3	1	AVX512_4FMAPS	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
EDX	4	4	1	Fast Short REP MOV	N/A		Native	
EDX	5	5	1	ULI	XFAM	XFAM[14]	As Configured (if Native)	
EDX	6	6	1	Reserved	N/A		Fixed	0x0
EDX	7	7	1	Reserved	N/A		Fixed	0x0
EDX	8	8	1	AVX512_VP2INTERSECT	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
EDX	9	9	1	Reserved	N/A		Fixed	0x0
EDX	10	10	1	MD_CLEAR supported	N/A		Native	
EDX	11	11	1	Reserved	N/A		Fixed	0x0
EDX	12	12	1	Reserved	N/A		Fixed	0x0
EDX	13	13	1	Reserved	N/A		Fixed	0x0
EDX	14	14	1	SERIALIZE Inst	N/A		Native	
EDX	15	15	1	Hetero Part	N/A		Native	
EDX	16	16	1	TSXLDTRK	N/A		Native	
EDX	17	17	1	Reserved	N/A		Fixed	0x0
EDX	18	18	1	PCONFIG	CPUID_CONFIG		As Configured (if Native)	
EDX	19	19	1	Architectural LBR support	XFAM	XFAM[15]	As Configured (if Native)	
EDX	20	20	1	CET	XFAM	XFAM[12:11]	As Configured (if Native)	
EDX	21	21	1	Reserved	N/A		Fixed	0x0
EDX	22	22	1	TMUL_AMX-BF16	XFAM	XFAM[18:17]	As Configured (if Native)	
EDX	23	23	1	FP16	XFAM	XFAM[7:5]	As Configured (if Native)	
EDX	24	24	1	TMUL_AMX-TILE	XFAM	XFAM[18:17]	As Configured (if Native)	
EDX	25	25	1	TMUL_AMX-INT8	XFAM	XFAM[18:17]	As Configured (if Native)	
EDX	26	26	1	IBRS (indirect branch restricted speculation)	N/A		Fixed	0x1
EDX	27	27	1	STIBP (single thread indirect branch predictors)	N/A		Native	
EDX	28	28	1	L1D_FLUSH. IA32_FLUSH_CMD support.	N/A		Native	
EDX	29	29	1	IA32_ARCH_CAPABILITIES Support	N/A		Fixed	0x1
EDX	30	30	1	IA32_CORE_CAPABILITIES Present	N/A		Fixed	0x1
EDX	31	31	1	SSBD (Speculative Store Bypass Disable)	N/A		Fixed	0x1
Leaf 0x7 / Sub-Leaf 0x1								
EAX	0	0	1	Reserved	N/A		Fixed	0x0
EAX	1	1	1	Reserved	N/A		Fixed	0x0
EAX	2	2	1	Reserved	N/A		Fixed	0x0
EAX	3	3	1	Reserved	N/A		Fixed	0x0
EAX	4	4	1	VEX VNNI	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
EAX	5	5	1	AVX512_BF16	XFAM, CPUID_CONFIG	XFAM[7:5]	As Configured (if Native)	
EAX	6	6	1	LASS	CPUID_CONFIG		As Configured (if Native)	
EAX	7	7	1	Reserved	N/A		Fixed	0x0
EAX	8	8	1	Reserved	N/A		Fixed	0x0
EAX	9	9	1	Reserved	N/A		Fixed	0x0
EAX	10	10	1	Fast Zero-Length MOVSB	N/A		Native	
EAX	11	11	1	Fast Short STOSB	N/A		Native	
EAX	12	12	1	Fast short CMPSB/SCASB	N/A		Native	
EAX	21	13	9	Reserved	N/A		Fixed	0x0
EAX	22	22	1	HRESET	N/A		Fixed	0x0
EAX	23	23	1	Reserved	N/A		Fixed	0x0
EAX	24	24	1	Reserved	N/A		Fixed	0x0
EAX	25	25	1	Reserved	N/A		Fixed	0x0
EAX	26	26	1	LAM	CPUID_CONFIG		As Configured (if Native)	
EAX	31	27	5	Reserved	N/A		Fixed	0x0

CPUID Field					Configuration by TDH.MNG.INIT		Virtualization	
Reg.	MSB	LSB	Field Size	Field Name	TD_PARAMS Section(s)	Configuration Details	Virtualization Type	Virtualization Details
EBX	29	0	30	Reserved	N/A		Fixed	0x0
EBX	30	30	1	Reserved	N/A		Fixed	0x0
EBX	31	31	1	Reserved	N/A		Fixed	0x0
ECX	31	0	32	Reserved	N/A		Fixed	0x0
EDX	0	0	1	Reserved	N/A		Fixed	0x0
EDX	1	1	1	Reserved	N/A		Fixed	0x0
EDX	2	2	1	Reserved	N/A		Fixed	0x0
EDX	3	3	1	Reserved	N/A		Fixed	0x0
EDX	4	4	1	Reserved	N/A		Fixed	0x0
EDX	5	5	1	Reserved	N/A		Fixed	0x0
EDX	31	6	26	Reserved	N/A		Fixed	0x0
				Leaf 0x8				
EAX	31	0	32	Reserved	N/A		Fixed	0x0
EBX	31	0	32	Reserved	N/A		Fixed	0x0
ECX	31	0	32	Reserved	N/A		Fixed	0x0
EDX	31	0	32	Reserved	N/A		Fixed	0x0
				Leaf 0xa				
EAX	7	0	8	Version	ATTRIBUTES	PERFMON	As Configured (if Native)	
EAX	15	8	8	Number of GP Counters	ATTRIBUTES	PERFMON	As Configured (if Native)	
EAX	23	16	8	Width of GP Counters	ATTRIBUTES	PERFMON	As Configured (if Native)	
EAX	31	24	8	Length of EBX Vector	ATTRIBUTES	PERFMON	As Configured (if Native)	
EBX	0	0	1	Core Cycles Not Available	ATTRIBUTES	PERFMON	As Configured (if Native)	
EBX	1	1	1	Instructions Retired Not Available	ATTRIBUTES	PERFMON	As Configured (if Native)	
EBX	2	2	1	Reference Cycles Not Available	ATTRIBUTES	PERFMON	As Configured (if Native)	
EBX	3	3	1	Last-Level Cache References Not Available	ATTRIBUTES	PERFMON	As Configured (if Native)	
EBX	4	4	1	Last-Level Cache Misses Not Available	ATTRIBUTES	PERFMON	As Configured (if Native)	
EBX	5	5	1	Branch Instruction Retired Not Available	ATTRIBUTES	PERFMON	As Configured (if Native)	
EBX	6	6	1	Branch Mispredict Retired Not Available	ATTRIBUTES	PERFMON	As Configured (if Native)	
EBX	31	7	25	Reserved	N/A		Fixed	0x0
ECX	31	0	32	Fixed Counter Support Bitmap	ATTRIBUTES	PERFMON	As Configured (if Native)	
EDX	4	0	5	Number of Fixed-Function Counters	ATTRIBUTES	PERFMON	As Configured (if Native)	
EDX	12	5	8	Width of Fixed-Function Counters	ATTRIBUTES	PERFMON	As Configured (if Native)	
EDX	13	13	1	Reserved	N/A		Fixed	0x0
EDX	14	14	1	Reserved	N/A		Fixed	0x0
EDX	15	15	1	AnyThread Deprecation	ATTRIBUTES	PERFMON	As Configured (if Native)	
EDX	31	16	16	Reserved	N/A		Fixed	0x0
				Leaf 0xd / Sub-Leaf 0x0				
EAX	0	0	1	X87	N/A		Fixed	0x1
EAX	1	1	1	SSE	N/A		Fixed	0x1
EAX	2	2	1	AVX256	XFAM	XFAM[2]	As Configured (if Native)	
EAX	3	3	1	PL_BNDREGS	N/A		Fixed	0x0
EAX	4	4	1	PL_BNDCFS	N/A		Fixed	0x0
EAX	5	5	1	KMASK	XFAM	XFAM[7:5]	As Configured (if Native)	
EAX	6	6	1	AVX3 ZMM 15:0	XFAM	XFAM[7:5]	As Configured (if Native)	
EAX	7	7	1	AVX3 ZMM 31:18	XFAM	XFAM[7:5]	As Configured (if Native)	
EAX	8	8	1	Reserved	N/A		Fixed	0x0
EAX	9	9	1	PKRU	XFAM	XFAM[9]	As Configured (if Native)	
EAX	16	10	7	Reserved	N/A		Fixed	0x0
EAX	17	17	1	AMX - XTILECFG	XFAM	XFAM[18:17]	As Configured (if Native)	
EAX	18	18	1	AMX - XTILEDATA	XFAM	XFAM[18:17]	As Configured (if Native)	
EAX	31	19	13	Reserved	N/A		Fixed	0x0
EBX	31	0	32	Max Bytes for Enabled Features	N/A		Calculated	Native
ECX	31	0	32	Max Bytes for Supported Features	XFAM		As Configured	
EDX	31	0	32	Reserved	N/A		Fixed	0x0
				Leaf 0xd / Sub-Leaf 0x1				
EAX	0	0	1	Supports XSAVEOPT	N/A		Fixed	0x1
EAX	1	1	1	Supports XSAVEC and compacted XRSTOR	N/A		Fixed	0x1
EAX	2	2	1	Supports XGETBV with ECX = 1	N/A		Native	
EAX	3	3	1	Supports XSAVES/XRSTORS and IA32_XSS	N/A		Fixed	0x1
EAX	4	4	1	XFD support	XFAM		As Configured	
EAX	31	5	27	Reserved	N/A		Fixed	0x0
EBX	31	0	32	Max Bytes for Enabled Features	N/A		Calculated	Native
ECX	7	0	8	Reserved	N/A		Fixed	0x0
ECX	8	8	1	XSS_RTIT	XFAM	XFAM[8]	As Configured (if Native)	
ECX	9	9	1	Reserved	N/A		Fixed	0x0
ECX	10	10	1	PASID	N/A		Fixed	0x0
ECX	11	11	1	U_CET	XFAM	XFAM[12:11]	As Configured (if Native)	
ECX	12	12	1	S_CET	XFAM	XFAM[12:11]	As Configured (if Native)	

CPUID Field					Configuration by TDH.MNG.INIT		Virtualization	
Reg.	MSB	LSB	Field Size	Field Name	TD_PARAMS Section(s)	Configuration Details	Virtualization Type	Virtualization Details
ECX	13	13	1	HDC	N/A		Fixed	0x0
ECX	14	14	1	ULI/UNIT	XFAM	XFAM[14]	As Configured (if Native)	
ECX	15	15	1	XSS_ARCH_LBRS	XFAM	XFAM[15]	As Configured (if Native)	
ECX	16	16	1	HWP Request	N/A		Fixed	0x0
ECX	31	17	15	Reserved	N/A		Fixed	0x0
EDX	31	0	32	Reserved	N/A		Fixed	0x0
				Leaf 0xd / Sub-Leaves 0x2-0x12				
EAX	31	0	32	Size	XFAM	XFAM[n]	As Configured (if Native)	
EBX	31	0	32	Offset	XFAM	XFAM[n]	As Configured (if Native)	
ECX	0	0	1	IA32_XSS	XFAM	XFAM[n]	As Configured (if Native)	
ECX	1	1	1		XFAM	XFAM[n]	As Configured (if Native)	
ECX	31	2	30		XFAM	XFAM[n]	As Configured (if Native)	
EDX	31	0	32	Reserved	N/A		Fixed	0x0
				Leaf 0xe				
EAX	31	0	32	Reserved	N/A		Fixed	0x0
EBX	31	0	32	Reserved	N/A		Fixed	0x0
ECX	31	0	32	Reserved	N/A		Fixed	0x0
EDX	31	0	32	Reserved	N/A		Fixed	0x0
				Leaf 0x11 / Sub-Leaf N/A				
EAX	31	0	32	Reserved	N/A		Fixed	0x0
EBX	31	0	32	Reserved	N/A		Fixed	0x0
ECX	31	0	32	Reserved	N/A		Fixed	0x0
EDX	31	0	32	Reserved	N/A		Fixed	0x0
				Leaf 0x12				
EAX	31	0	32	Reserved	N/A		Fixed	0x0
EBX	31	0	32	Reserved	N/A		Fixed	0x0
ECX	31	0	32	Reserved	N/A		Fixed	0x0
EDX	31	0	32	Reserved	N/A		Fixed	0x0
				Leaf 0x13				
EAX	31	0	32	Reserved	N/A		Fixed	0x0
EBX	31	0	32	Reserved	N/A		Fixed	0x0
ECX	31	0	32	Reserved	N/A		Fixed	0x0
EDX	31	0	32	Reserved	N/A		Fixed	0x0
				Leaf 0x14 / Sub-Leaf 0x0				
EAX	31	0	32	Max Valid Subleaf	XFAM	XFAM[8]	As Configured (if Native)	
EBX	0	0	1	CR3 Filtering	XFAM	XFAM[8]	As Configured (if Native)	
EBX	1	1	1	Cycle Accurate Mode	XFAM	XFAM[8]	As Configured (if Native)	
EBX	2	2	1	IP Filtering	XFAM	XFAM[8]	As Configured (if Native)	
EBX	3	3	1	MSRs Preserved Across Warm Reset	XFAM	XFAM[8]	As Configured (if Native)	
EBX	4	4	1	PTWRITE Support	XFAM	XFAM[8]	As Configured (if Native)	
EBX	5	5	1	Power Event Trace Support	XFAM	XFAM[8]	As Configured (if Native)	
EBX	6	6	1	PSB/PMI Injection Support	XFAM	XFAM[8]	As Configured (if Native)	
EBX	7	7	1	PT Event Trace Support	XFAM	XFAM[8]	As Configured (if Native)	
EBX	31	8	24	Reserved	XFAM	XFAM[8]	As Configured (if Native)	
ECX	0	0	1	ToPA Output Supported	XFAM	XFAM[8]	As Configured (if Native)	
ECX	1	1	1	ToPA Tables Support Multiple Regions	XFAM	XFAM[8]	As Configured (if Native)	
ECX	2	2	1	Single-Range Output Supported	XFAM	XFAM[8]	As Configured (if Native)	
ECX	30	3	28	Reserved	XFAM	XFAM[8]	As Configured (if Native)	
ECX	31	31	1	IP Payload Contains LIP	XFAM	XFAM[8]	As Configured (if Native)	
EDX	31	0	32	Reserved	XFAM	XFAM[8]	As Configured (if Native)	
				Leaf 0x14 / Sub-Leaf 0x1				
EAX	1	0	2	MTC Period Options	XFAM	XFAM[8]	As Configured (if Native)	
EAX	15	2	14	Reserved	XFAM	XFAM[8]	As Configured (if Native)	
EAX	31	16	16	Number of Address Ranges Supported	XFAM	XFAM[8]	As Configured (if Native)	
EBX	15	0	16	Cycle Thresholds	XFAM	XFAM[8]	As Configured (if Native)	
EBX	31	16	16	PSB Frequencies	XFAM	XFAM[8]	As Configured (if Native)	
ECX	31	0	32	Reserved	XFAM	XFAM[8]	As Configured (if Native)	
EDX	31	0	32	Reserved	XFAM	XFAM[8]	As Configured (if Native)	
				Leaf 0x15				
EAX	31	0	32	Denominator	N/A		Fixed	0x1
EBX	31	0	32	Numerator	Other	TSC_FREQUENCY	As Configured	
ECX	31	0	32	Nominal ART Frequency	N/A		Fixed	0x017D7840
EDX	31	0	32	Reserved	N/A		Fixed	0x0
				Leaf 0x19				
EAX	0	0	1	CPL0 Restriction	ATTRIBUTES	KL	As Configured (if Native)	
EAX	1	1	1	Decrypt only Restriction	ATTRIBUTES	KL	As Configured (if Native)	
EAX	2	2	1	Encrypt only Restriction	ATTRIBUTES	KL	As Configured (if Native)	
EAX	3	3	1	Process Restriction	ATTRIBUTES	KL	As Configured (if Native)	

CPUID Field					Configuration by TDH.MNG.INIT		Virtualization	
Reg.	MSB	LSB	Field Size	Field Name	TD_PARAMS Section(s)	Configuration Details	Virtualization Type	Virtualization Details
EAX	31	4	28	Reserved_31_4	ATTRIBUTES	KL	As Configured (if Native)	
EBX	0	0	1	AES KL Enabled	ATTRIBUTES	KL	As Configured (if Native)	
EBX	1	1	1	Reserved	ATTRIBUTES	KL	As Configured (if Native)	
EBX	2	2	1	AES wide KL Support	ATTRIBUTES	KL	As Configured (if Native)	
EBX	3	3	1	Reserved	ATTRIBUTES	KL	As Configured (if Native)	
EBX	4	4	1	IW Key Backup Support	ATTRIBUTES	KL	As Configured (if Native)	
EBX	31	5	27	Reserved	ATTRIBUTES	KL	As Configured (if Native)	
ECX	0	0	1	LOADIWKEY No Backup parameter Support	ATTRIBUTES	KL	As Configured (if Native)	
ECX	1	1	1	Random IWKey Support	N/A		Fixed	0x0
ECX	31	2	30	Reserved	N/A		Fixed	0x0
EDX	31	0	32	Reserved	N/A		Fixed	0x0
				Leaf 0x1c				
EAX	7	0	8	Supported LBR depth values	XFAM, CPUID_CONFIG	XFAM[15]	As Configured (if Native)	
EAX	29	8	22	Reserved_29_8	XFAM	XFAM[15]	As Configured (if Native)	
EAX	30	30	1	Deep C-state May Reset	XFAM	XFAM[15]	As Configured (if Native)	
EAX	31	31	1	IP values contain LIP	XFAM	XFAM[15]	As Configured (if Native)	
EBX	0	0	1	CPL Filtering Supported	XFAM	XFAM[15]	As Configured (if Native)	
EBX	1	1	1	Branch Filtering Supported	XFAM	XFAM[15]	As Configured (if Native)	
EBX	2	2	1	Call-stack Mode Supported	XFAM	XFAM[15]	As Configured (if Native)	
EBX	31	3	29	Reserved_31_3	XFAM	XFAM[15]	As Configured (if Native)	
ECX	0	0	1	Mispredict Bit Supported	XFAM	XFAM[15]	As Configured (if Native)	
ECX	1	1	1	Timed LBRs Supported	XFAM	XFAM[15]	As Configured (if Native)	
ECX	2	2	1	Branch Type Field Supported	XFAM	XFAM[15]	As Configured (if Native)	
ECX	31	3	29	Reserved_31_3	XFAM	XFAM[15]	As Configured (if Native)	
EDX	31	0	32	Reserved	XFAM	XFAM[15]	As Configured (if Native)	
				Leaf 0x1d / Sub-Leaf 0x0				
EAX	0	0	1	TILE support	XFAM	XFAM[18:17]	As Configured (if Native)	
EAX	31	1	31	Reserved_31_1	XFAM	XFAM[18:17]	As Configured (if Native)	
EBX	31	0	32	Reserved	XFAM	XFAM[18:17]	As Configured (if Native)	
ECX	31	0	32	Reserved	XFAM	XFAM[18:17]	As Configured (if Native)	
EDX	31	0	32	Reserved	XFAM	XFAM[18:17]	As Configured (if Native)	
				Leaf 0x1d / Sub-Leaf 0x1				
EAX	15	0	16	total_tile_bytes	XFAM	XFAM[18:17]	As Configured (if Native)	
EAX	31	16	16	bytes_per_tile	XFAM	XFAM[18:17]	As Configured (if Native)	
EBX	15	0	16	bytes_per_row	XFAM	XFAM[18:17]	As Configured (if Native)	
EBX	31	16	16	max_names	XFAM	XFAM[18:17]	As Configured (if Native)	
ECX	15	0	16	max_rows	XFAM	XFAM[18:17]	As Configured (if Native)	
ECX	31	16	16	Reserved_31_16	XFAM	XFAM[18:17]	As Configured (if Native)	
EDX	31	0	32	Reserved	XFAM	XFAM[18:17]	As Configured (if Native)	
				Leaf 0x1e / Sub-Leaf N/A				
EAX	31	0	32	Reserved	XFAM	XFAM[18:17]	As Configured (if Native)	
EBX	7	0	8	impl.tmul_maxk (rows or cols)	XFAM	XFAM[18:17]	As Configured (if Native)	
EBX	23	8	16	impl.tmul_maxn (column bytes)	XFAM	XFAM[18:17]	As Configured (if Native)	
EBX	31	24	8	Reserved_31_24	XFAM	XFAM[18:17]	As Configured (if Native)	
ECX	31	0	32	Reserved	XFAM	XFAM[18:17]	As Configured (if Native)	
EDX	31	0	32	Reserved	XFAM	XFAM[18:17]	As Configured (if Native)	
				Leaf 0x20				
EAX	31	0	32	Reserved	N/A		Fixed	0x0
EBX	31	0	32	Reserved	N/A		Fixed	0x0
ECX	31	0	32	Reserved	N/A		Fixed	0x0
EDX	31	0	32	Reserved	N/A		Fixed	0x0
				Leaf 0x21 / Sub-Leaf 0x0				
EAX	31	0	32	Maximum sub-leaf	N/A		Fixed	0x00000000
EBX	31	0	32	“Inte”	N/A		Fixed	0x65746E49
ECX	31	0	32	“ “	N/A		Fixed	0x20202020
EDX	31	0	32	“ITDX”	N/A		Fixed	0x5844546C
				Leaf 0x80000000				
EAX	31	0	32	MaxIndex	N/A		Native	
EBX	31	0	32	Reserved	N/A		Fixed	0x0
ECX	31	0	32	Reserved	N/A		Fixed	0x0
EDX	31	0	32	Reserved	N/A		Fixed	0x0
				Leaf 0x80000001				
EAX	31	0	32	Reserved	N/A		Fixed	0x0
EBX	31	0	32	Reserved	N/A		Fixed	0x0
ECX	0	0	1	LAHF/SAHF in 64-bit Mode	N/A		Native	
ECX	4	1	4	Reserved_4_1	N/A		Fixed	0x0
ECX	5	5	1	LZCNT	N/A		Native	

CPUID Field					Configuration by TDH.MNG.INIT		Virtualization	
Reg.	MSB	LSB	Field Size	Field Name	TD_PARAMS Section(s)	Configuration Details	Virtualization Type	Virtualization Details
ECX	7	6	2	Reserved_7_6	N/A		Fixed	0x0
ECX	8	8	1	PREFETCHW	N/A		Native	
ECX	31	9	23	Reserved_31_9	N/A		Fixed	0x0
EDX	10	0	11	Reserved_10_0	N/A		Fixed	0x0
EDX	11	11	1	SYSCALL/SYSRET in 64-bit Mode	N/A		Native	
EDX	19	12	8	Reserved_19_12	N/A		Fixed	0x0
EDX	20	20	1	Execute Disable Bit	N/A		Fixed	0x1
EDX	25	21	5	Reserved_25_21	N/A		Fixed	0x0
EDX	26	26	1	1GB Pages	N/A		Fixed	0x1
EDX	27	27	1	RDTSCP and IA32_TSC_AUX	N/A		Fixed	0x1
EDX	28	28	1	Reserved_28	N/A		Fixed	0x0
EDX	29	29	1	Intel 64	N/A		Fixed	0x1
EDX	31	30	2	Reserved_31_30	N/A		Fixed	0x0
Leaf 0x80000008								
EAX	7	0	8	Number of Physical Address Bits	N/A		Fixed	0x34
EAX	15	8	8	Number of Linear Address Bits	N/A		Native	
EAX	31	16	16	Reserved	N/A		Fixed	0x0
EBX	8	0	9	Reserved_8_0	N/A		Fixed	0x0
EBX	9	9	1	WBNOINVD support	N/A		Fixed	0x1
EBX	31	10	22	Reserved_31_10	N/A		Fixed	0x0
ECX	31	0	32	Reserved	N/A		Fixed	0x0
EDX	31	0	32	Reserved	N/A		Fixed	0x0

3. ABI Reference: Constants

This chapter describes the constants designed to be used in the Intel TDX module.

3.1. Interface Function Completion Status Codes

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

This section defines the function completion status codes. The structure of the status codes is described in the [TDX Module Spec]. Three tables are provided below: class table, code table and operand ID table.

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

Table 3.1: 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

3.1.2. Function Completion Status Codes

Table 3.2: Function Completion Status Codes Definition

Flags, Class and Name (Bits 63:32)			Details L2 (Bits 31:0)	Description
Value (Hex)	Status	Name		
0x00000000	Success	TDX_SUCCESS	For TDH.VP.ENTER: Exit Reason For list operations: Number of problematic sub-operations (e.g., a page in a list not processed due to an incorrect state).	Function completed successfully.
0x40000001	Non-Recover.	TDX_NON_RECOVERABLE_VCPU	For TDH.VP.ENTER: Exit Reason	TD exit due to a non-recoverable VCPU state (e.g., triple fault) – VCPU is disabled
0x60000002	Fatal	TDX_NON_RECOVERABLE_TD	For TDH.VP.ENTER: Exit Reason	TD exit due to a non-recoverable TD state – TD is disabled

Flags, Class and Name (Bits 63:32)			Details L2 (Bits 31:0)	Description
Value (Hex)	Status	Name		
0x80000003	Recover. Error	TDX_INTERRUPTED_RESUMABLE	For list operations: Number of problematic sub-operations (e.g., a page in a list not processed due to an incorrect state).	Function operation has been interrupted by an external event, and it may be resumed from the point it was interrupted by calling it again.
0x80000004	Recover. Error	TDX_INTERRUPTED_RESTARTABLE	For list operations: Number of problematic sub-operations (e.g., a page in a list not processed due to an incorrect state).	Function operation has been interrupted by an external event, and it may be restarted (from its beginning) by calling it again.
0x60000005	Fatal	TDX_NON_RECOVERABLE_TD_NON_ACCESSIBLE	For TDH.VP.ENTER: Exit Reason	TD exit due to a fatal TD state (e.g., machine check caused by a memory integrity check error) – TD is disabled and its private memory can't be accessed.
0xC0000006	Error	TDX_INVALID_RESUMPTION	0	Resumed function in invalid, e.g., its operands are different than the last interrupted function.
0xE0000007	Fatal	TDX_NON_RECOVERABLE_TD_NO_APIC	TDH.VP.ENTER: Exit Reason	TD is running with local APIC disabled
0x00000008	Success	TDX_CROSS_TD_FAULT	0	Fault-like TD exit due to a cross-TD error, i.e., the current TD encountered an error that is related to some other TD.
0x10000009	Success	TDX_CROSS_TD_TRAP	0	Trap-like TD exit due to a cross-TD error, i.e., the current TD encountered an error that is related to some other TD.
0x6000000A	Fatal	TDX_NON_RECOVERABLE_TD_CORRUPTED_MD	0	TD exit due to a non-recoverable corrupted TD metadata – TD is disabled
0xC0000100	Error	TDX_OPERAND_INVALID	Operand ID	Operand is invalid.
0xC0000101	Error	TDX_OPERAND_ADDR_RANGE_ERROR	Operand ID	Operand address is out of range (e.g., not in a TDMR).
0x80000200	Recover. Error	TDX_OPERAND_BUSY	Operand ID	The operand is busy (e.g., it is locked in Exclusive mode).
0x80000201	Recover. Error	TDX_PREVIOUS_TLB_EPOCH_BUSY	0	TDH.MEM.TRACK failed because one or more of the TD's VCPUs are running, and their VCPU epoch is the previous TD epoch.
0x80000202	Recover. Error	TDX_SYS_BUSY	0	The Intel TDX module (as a whole) is busy.
0xC0000300	Error	TDX_PAGE_METADATA_INCORRECT	Operand ID	Physical page metadata (in PAMT) are incorrect for the requested operation.
0x00000301	Success	TDX_PAGE_ALREADY_FREE	Operand ID	Physical page is already marked as PT_FREE.
0xC0000302	Error	TDX_PAGE_NOT_OWNED_BY_TD	Operand ID	Physical page PAMT entry's OWNER field does not point to the TD's TDR page
0xC0000303	Error	TDX_PAGE_NOT_FREE	Operand ID	Physical page is not free
0xC0000400	Error	TDX_TD_ASSOCIATED_PAGES_EXIST	0	Physical pages associated with the TD exist in memory.
0xC0000500	Error	TDX_SYSINIT_NOT_PENDING	0	Attempting TDH.SYS.INIT when not expected.
0xC0000501	Error	TDX_SYSINIT_NOT_DONE	0	Attempting non-TDH.SYS.INIT SEAMCALL leaf before TDH.SYS.INIT was done.
0xC0000502	Error	TDX_SYSINITLP_NOT_DONE	0	Attempting non-TDH.SYS.LP.INIT SEAMCALL leaf before TDH.SYS.LP.INIT was done on this LP.
0xC0000503	Error	TDX_SYSINITLP_DONE	0	Attempting TDH.SYS.LP.INIT when already done on this LP.
0xC0000505	Error	TDX_SYS_NOT_READY	0	Attempting to execute a non-initialization SEAMCALL function before initialization sequence completed.
0xC0000506	Error	TDX_SYS_SHUTDOWN	0	Attempting to execute SEAMCALL when the Intel TDX module is being shut down.
0xC0000507	Error	TDX_SYSCONFIG_NOT_DONE	0	Attempting TDH.SYS.KEY.CONFIG before TDH.SYS.CONFIG is done.
0xC0000600	Error	TDX_TD_NOT_INITIALIZED	Operand ID (0 if default)	TD has not been initialized (by TDH.MNG.INIT).
0xC0000601	Error	TDX_TD_INITIALIZED	Operand ID (0 if default)	TD has been initialized (by TDH.MNG.INIT).
0xC0000602	Error	TDX_TD_NOT_FINALIZED	Operand ID (0 if default)	TD measurement has not been finalized (by TDH.MR.FINALIZE).
0xC0000603	Error	TDX_TD_FINALIZED	Operand ID (0 if default)	TD measurement has been finalized (by TDH.MR.FINALIZE).
0xC0000604	Error	TDX_TD_FATAL	Operand ID (0 if default)	TD is in a FATAL error state.
0xC0000605	Error	TDX_TD_NON_DEBUG	Operand ID (0 if default)	TD's ATTRIBUTES.DEBUG bit is 0.
0xC0000606	Error	TDX_TDCS_NOT_ALLOCATED	Operand ID (0 if default)	TDCS pages have not been allocated
0xC0000607	Error	TDX_LIFECYCLE_STATE_INCORRECT	Operand ID (0 if default)	The TD's LIFECYCLE_STATE is incorrect for the required operation.
0xC0000608	Error	TDX_OP_STATE_INCORRECT	Operand ID (0 if default)	The TD's OP_STATE is incorrect for the required operation.
0xC0000610	Error	TDX_TDCX_NUM_INCORRECT	Operand ID (0 if default)	The number of TDCX pages is incorrect.
0xC0000700	Error	TDX_VCPU_STATE_INCORRECT	0	The VCPU state is incorrect for the requested operation.
0x80000701	Recover. Error	TDX_VCPU_ASSOCIATED	0	The VCPU is already associated with another LP.

Flags, Class and Name (Bits 63:32)			Details L2 (Bits 31:0)	Description
Value (Hex)	Status	Name		
0x80000702	Recover. Error	TDX_VCPU_NOT_ASSOCIATED	0	The VCPU is not associated with the current LP.
0xC0000704	Error	TDX_NO_VALID_VE_INFO	0	There is no valid #VE information.
0xC0000705	Error	TDX_MAX_VCPUS_EXCEEDED	0	TD's maximum number of VCPUs has been exceeded.
0xC0000706	Error	TDX_TSC_ROLLBACK	0	Time Stamp Counter value is lower than on last TD exit.
0xC0000730	Error	TDX_TD_VMCS_FIELD_NOT_INITIALIZED	Bits 31:0: VMCS field code	The TD VMCS field has not been initialized.
0x80000800	Recover. Error	TDX_KEY_GENERATION_FAILED	0	Failed to generate a random key.
0x80000810	Recover. Error	TDX_TD_KEYS_NOT_CONFIGURED	0	TD keys have not been configured on the hardware.
0xC0000811	Error	TDX_KEY_STATE_INCORRECT	0	KOT entry state is incorrect for the required operation.
0x00000815	Success	TDX_KEY_CONFIGURED	0	The key is already configured on the current package.
0x80000817	Recover. Error	TDX_WBCACHE_NOT_COMPLETE	0	Attempting to execute TDH.MNG.KEY.FREEID when TDH.PHYMEM.CACHE.WB has not completed its operation.
0xC0000820	Error	TDX_HKID_NOT_FREE	0	A provided HKID cannot be assigned because it is not free.
0x00000821	Success	TDX_NO_HKID_READY_TO_WBCACHE	0	No private HKID is in the HKID_FLUSHED state, ready for TDH.PHYMEM.CACHE.WB.
0xC0000823	Error	TDX_WBCACHE_RESUME_ERROR	0	Resume of a previously interrupted function has been aborted due to wrong HKID.
0x80000824	Recover. Error	TDX_FLUSHVP_NOT_DONE	0	TDH.VP.FLUSH was not done on all required VCPUs; some VCPUs are still associated with LPs.
0xC0000825	Error	TDX_NUM_ACTIVATED_HKIDS_NOT_SUPPORTED	Bits 31:0: Maximum supported HKIDs	The number of activated key IDs on the platform is not supported.
0xC0000900	Error	TDX_INCORRECT_CPUID_VALUE	0	A CPUID value is incorrect.
0xC0000901	Error	TDX_BOOT_NT4_SET	0	MSR IA32_MISC_ENABLES bit 22 (Boot NT4) is set.
0xC0000902	Error	TDX_INCONSISTENT_CPUID_FIELD	0	A field returned by CPUID is inconsistent between LPs.
0xC0000904	Error	TDX_CPUID_LEAF_1F_FORMAT_UNRECOGNIZED	0	CPUID leaf 1F format is not recognized or sub-leaves are not in order.
0xC0000905	Error	TDX_INVALID_WBINVD_SCOPE	0	WBINVD scope is not supported.
0xC0000906	Error	TDX_INVALID_PKG_ID	Package ID	Package ID is larger than the maximum supported.
0xC0000908	Error	TDX_CPUID_LEAF_NOT_SUPPORTED	CPUID leaf	0
0xC0000910	Error	TDX_SMRR_NOT_LOCKED	0: SMRR, 1: SMRR2	SMRR* is not locked.
0xC0000911	Error	TDX_INVALID_SMRR_CONFIGURATION	0: SMRR, 1: SMRR2	SMRR* configuration is invalid.
0xC0000912	Error	TDX_SMRR_OVERLAPS_CMR	Bits 7:0: 0: SMRR, 1: SMRR2 Bits 15:8: Overlapping CMR index	SMRR* overlaps a CMR.
0xC0000913	Error	TDX_SMRR_LOCK_NOT_SUPPORTED	0	Platform does not support SMRR locking.
0xC0000914	Error	TDX_SMRR_NOT_SUPPORTED	0	Platform does not support SMRR.
0xC0000920	Error	TDX_INCONSISTENT_MSR	Bits 31:0: MSR index	MSR configuration is inconsistent between LPs.
0xC0000921	Error	TDX_INCORRECT_MSR_VALUE	Bits 31:0: MSR index	MSR value is incorrect.
0xC0000930	Error	TDX_SEAMREPORT_NOT_AVAILABLE	0	SEAMOPS(SEAMREPORT) instruction leaf is not available.
0xC0000A00	Error	TDX_INVALID_TDMR	Bits 7:0: TDMR index	TDMR base address is not aligned on 1GB, its HKID bits are not 0, TDMR size is not specified with 1GB granularity or TDMR is outside the platform's maximum PA.
0xC0000A01	Error	TDX_NON_ORDERED_TDMR	Bits 7:0: TDMR index	TDMR is not specified in an ascending, non-overlapping order.
0xC0000A02	Error	TDX_TDMR_OUTSIDE_CMRS	Bits 7:0: TDMR index	TDMR non-reserved parts are not fully contained in CMRs.
0x00000A03	Success	TDX_TDMR_ALREADY_INITIALIZED	0	TDMR is already fully initialized.
0xC0000A10	Error	TDX_INVALID_PAMT	Bits 7:0: TDMR index Bits 15:8: PAMT level (2: 1GB, 1: 2MB, 0: 4KB)	PAMT region base address is not aligned on 4KB, its HKID bits are not 0, PAMT region size is not specified with 4KB granularity, it is not large enough for the TDMR size or PAMT region is outside the platform's maximum PA.
0xC0000A11	Error	TDX_PAMT_OUTSIDE_CMRS	Bits 7:0: TDMR index Bits 15:8: PAMT level (2: 1GB, 1: 2MB, 0: 4KB)	PAMT is not fully contained in CMRs.
0xC0000A12	Error	TDX_PAMT_OVERLAP	Bits 7:0: TDMR index Bits 15:8: PAMT level (2: 1GB, 1: 2MB, 0: 4KB) Bits 23:16: Overlapping TDMR index	PAMT overlaps with TDMR non-reserved parts or with another PAMT.
0xC0000A20	Error	TDX_INVALID_RESERVED_IN_TDMR	Bits 7:0: TDMR index Bits 15:8: Reserved area index	Reserved area in TMDR's base offset is not aligned on 4KB, its size is not specified with 4KB granularity or it is not fully contained within the TDMR.
0xC0000A21	Error	TDX_NON_ORDERED_RESERVED_IN_TDMR	Bits 7:0: TDMR index Bits 15:8: Reserved area index	Reserved area in TDMR is not specified in an ascending, non-overlapping order.

Flags, Class and Name (Bits 63:32)			Details L2 (Bits 31:0)	Description
Value (Hex)	Status	Name		
0xC0000A22	Error	TDX_CMR_LIST_INVALID	0	CMR list provided to the TDX module is invalid
0xC0000B00	Error	TDX_EPT_WALK_FAILED	Operand ID	EPT walk failed
0xC0000B01	Error	TDX_EPT_ENTRY_FREE	Operand ID	EPT entry is free
0xC0000B02	Error	TDX_EPT_ENTRY_NOT_FREE	Operand ID	EPT entry is not free
0xC0000B03	Error	TDX_EPT_ENTRY_NOT_PRESENT	Operand ID	EPT entry is not present
0xC0000B04	Error	TDX_EPT_ENTRY_NOT_LEAF	Operand ID	EPT entry is not a leaf
0xC0000B05	Error	TDX_EPT_ENTRY_LEAF	Operand ID	EPT entry is a leaf
0xC0000B06	Error	TDX_GPA_RANGE_NOT_BLOCKED	Operand ID	GPA range is not blocked
0x00000B07	Success	TDX_GPA_RANGE_ALREADY_BLOCKED	Operand ID	GPA range is already blocked
0xC0000B08	Error	TDX_TLB_TRACKING_NOT_DONE	Operand ID	TLB tracking has not been done
0xC0000B09	Error	TDX_EPT_INVALID_PROMOTE_CONDITIONS	Operand ID	Conditions for GPA mapping promotions as invalid
0x00000B0A	Success	TDX_PAGE_ALREADY_ACCEPTED	Error EPT level	Page has already been accepted
0xC0000B0B	Error	TDX_PAGE_SIZE_MISMATCH	Error EPT level	Requested page size does not match the current GPA mapping size
0xC0000B0C	Error	TDX_GPA_RANGE_BLOCKED	Operand ID	GPA range is blocked
0xC0000B0D	Error	TDX_EPT_ENTRY_STATE_INCORRECT	Operand ID	EPT entry state is incorrect
0xC0000B0E	Error	TDX_EPT_PAGE_NOT_FREE	Operand ID	EPT page is not free
0xC0000C00	Error	TDX_METADATA_FIELD_ID_INCORRECT	For a single field, set to 0. For a metadata list: Bits 15:0: Sequence number 0xFFFF indicates the list header. Bits 31:16: Field number in sequence 0xFFFF indicates the sequence header.	The provided FIELD_ID is incorrect.
0xC0000C01	Error	TDX_METADATA_FIELD_NOT_WRITABLE	For a single field, set to 0. For a metadata list: Bits 15:0: Sequence number 0xFFFF indicates the list header. Bits 31:16: Field number in sequence 0xFFFF indicates the sequence header.	Field code and write mask are for a read-only field.
0xC0000C02	Error	TDX_METADATA_FIELD_NOT_READABLE	For a single field, set to 0. For a metadata list: Bits 15:0: Sequence number 0xFFFF indicates the list header. Bits 31:16: Field number in sequence 0xFFFF indicates the sequence header.	Field code is for an unreadable field.
0xC0000C03	Error	TDX_METADATA_FIELD_VALUE_NOT_VALID	For a single field, set to 0. For a metadata list: Bits 15:0: Sequence number 0xFFFF indicates the list header. Bits 31:16: Field number in sequence 0xFFFF indicates the sequence header.	The provided field value is not valid.
0xC0000C04	Error	TDX_METADATA_LIST_OVERFLOW	For a single field, set to 0. For a metadata list: Bits 15:0: Sequence number 0xFFFF indicates the list header. Bits 31:16: Field number in sequence 0xFFFF indicates the sequence header.	A metadata list does not fit within the provided buffer
0xC0000C05	Error	TDX_INVALID_METADATA_LIST_HEADER	0	Metadata list header is invalid
0xC0000C06	Error	TDX_REQUIRED_METADATA_FIELD_MISSING	0	A required metadata field is missing
0xC0000C07	Error	TDX_METADATA_ELEMENT_SIZE_INCORRECT	For a single field, set to 0. For a metadata list: Bits 15:0: Sequence number 0xFFFF indicates the list header. Bits 31:16: Field number in sequence 0xFFFF indicates the sequence header.	A metadata field identifier specifies an incorrect ELEMENT_SIZE_CODE for the field
0xC0000C08	Error	TDX_METADATA_LAST_ELEMENT_INCORRECT	For a single field, set to 0. For a metadata list: Bits 15:0: Sequence number 0xFFFF indicates the list header. Bits 31:16: Field number in sequence 0xFFFF indicates the sequence header.	A metadata field identifier specifies an incorrect LAST_ELEMENT_IN_FIELD for the field

Flags, Class and Name (Bits 63:32)			Details L2 (Bits 31:0)	Description
Value (Hex)	Status	Name		
0xC0000C09	Error	TDX_METADATA_FIELD_CURRENTLY_NOT_WRITABLE	For a single field, set to 0. For a metadata list: Bits 15:0: Sequence number 0xFFFF indicates the list header. Bits 31:16: Field number in sequence 0xFFFF indicates the sequence header.	The metadata field is currently not writable, e.g., per some state of the TD
0xC0000D00	Error	TDX_SERVTD_ALREADY_BOUND_FOR_TYPE	0	A single service TD of this type is supported
0xC0000D01	Error	TDX_SERVTD_TYPE_MISMATCH	0	Service TD type does not match the currently bound type
0xC0000D02	Error	TDX_SERVTD_ATTR_MISMATCH	0	Service TD attributes do not match the currently bound attributes
0xC0000D03	Error	TDX_SERVTD_INFO_HASH_MISMATCH	0	Service TD hash of TDINFO_STRUCT does not match the currently bound hash
0xC0000D04	Error	TDX_SERVTD_UUID_MISMATCH	0	Service TD UUID does not match the currently bound UUID
0xC0000D05	Error	TDX_SERVTD_NOT_BOUND	Binding slot number	Service TD is not bound
0xC0000D06	Error	TDX_SERVTD_BOUND	0	Service TD is already bound
0xC0000D07	Error	TDX_TARGET_UUID_MISMATCH	0	Taget TD UUID does not match the requested TD_UUID
0xC0000D08	Error	TDX_TARGET_UUID_UPDATED	0	Taget TD UUID does not match the requested TD_UUID, but pre-migration target TD UUID does match it
0xC0000E00	Error	TDX_INVALID_MBMD	0	MBMD is invalid
0xC0000E01	Error	TDX_INCORRECT_MBMD_MAC	0	MBMD.MAC field value is incorrect
0xC0000E02	Error	TDX_NOT_WRITE_BLOCKED	Operand ID	Secure EPT entry is not blocked for writing
0x00000E03	Success	TDX_ALREADY_WRITE_BLOCKED	Operand ID	Secure EPT entry is already blocked for writing
0xC0000E04	Error	TDX_NOT_EXPORTED	Operand ID	Secure EPT entry is not marked as exported
0xC0000E05	Error	TDX_MIGRATION_STREAM_STATE_INCORRECT	0	Migraion stream has not been initialized or is not enabled
0xC0000E06	Error	TDX_MAX_MIGS_NUM_EXCEEDED	MAX_MIGS	The maximum number of supported migration streams has been exceeded
0xC0000E07	Error	TDX_EXPORTED_DIRTY_PAGES_REMAIN	0	There are some pages that have been exported, but need to be re-exported because their contents have changed
0xC0000E08	Error	TDX_MIGRATION_SESSION_KEY_NOT_SET	0	A new migration session key has not been set before a migration session start is attempted
0xC0000E09	Error	TDX_TD_NOT_MIGRATABLE	0	The TD's ATTRIBUTES.MIGRATABLE bit is not set
0xC0000E0A	Error	TDX_PREVIOUS_EXPORT_CLEANUP_INCOMPLETE	0	A previous aborted export session cleanup (using TDH.EXPORT.CANCEL) has not been completed
0xC0000E0B	Error	TDX_NUM_MIGS_HIGHER_THAN_CREATED	0	The number of migration streams used by the session is higher than the number of created migration streams
0xC0000E0C	Error	TDX_IMPORT_MISMATCH	0	A re-import or an import cancellation does not match the existing Secure EPT entry
0xC0000E0D	Error	TDX_MIGRATION_EPOCH_OVERFLOW	0	Migration epoch has exceeded its maximum value
0xC0000E0E	Error	TDX_MAX_EXPORTS_EXCEEDED	0	Maximum number of TD export attempts (2^31) has been exceeded
0xC0000E0F	Error	TDX_INVALID_PAGE_MAC	Operand ID	Imported page MAC is invalid
0xC0000E10	Error	TDX_MIGRATED_IN_CURRENT_EPOCH	Operand ID	Page already migrated in the current epoch
0xC0000E11	Error	TDX_DISALLOWED_IMPORT_OVER_REMOVED	Operand ID	Disallowed age import over a previously-removed page

3.1.3. Function Completion Status Operand IDs

Table 3.3: Function Completion Operand IDs Definition

Operand ID	Explicit/ Implicit	Class	Operand	Description
0	Explicit	GPR	RAX	Explicit input operand RAX
1	Explicit	GPR	RCX	Explicit input operand RCX
2	Explicit	GPR	RDX	Explicit input operand RDX
3	Explicit	GPR	RBX	Explicit input operand RBX
5	Explicit	GPR	RBP	Explicit input operand RBP

Operand ID	Explicit/ Implicit	Class	Operand	Description
6	Explicit	GPR	RSI	Explicit input operand RSI
7	Explicit	GPR	RDI	Explicit input operand RDI
8	Explicit	GPR	R8	Explicit input operand R8
9	Explicit	GPR	R9	Explicit input operand R9
10	Explicit	GPR	R10	Explicit input operand R10
11	Explicit	GPR	R11	Explicit input operand R11
12	Explicit	GPR	R12	Explicit input operand R12
13	Explicit	GPR	R13	Explicit input operand R13
14	Explicit	GPR	R14	Explicit input operand R14
15	Explicit	GPR	R15	Explicit input operand R15
64	Explicit	Component of explicit input	ATTRIBUTES	TD_PARAMS.ATTRIBUTES
65	Explicit	Component of explicit input	XFAM	TD_PARAMS.XFAM
66	Explicit	Component of explicit input	EXEC_CONTROLS	TD_PARAMS.EXEC_CONTROLS
67	Explicit	Component of explicit input	EPTP_CONTROLS	TD_PARAMS.EPTP_CONTROLS
68	Explicit	Component of explicit input	MAX_VCPUS	TD_PARAMS.MAX_VCPUS
69	Explicit	Component of explicit input	CPUID_CONFIG	TD_PARAMS.CPUID_CONFIG
70	Explicit	Component of explicit input	TSC_FREQUENCY	TD_PARAMS.TSC_FREQUENCY
95	Explicit	Component of explicit input	PAGE	PAGE PA array entry
96	Explicit	Component of explicit input	TDMR_INFO_PA	TDMR_INFO_PA array entry
97	Explicit	Component of explicit input	GPA_LIST_ENTRY	GPA_LIST array entry
98	Explicit	Component of explicit input	MIG_BUFF_LIST_ENTRY	Migration buffer list entry
99	Explicit	Component of explicit input	NEW_PAGE_LIST_ENTRY	New page list entry
128	Implicit	Physical Page	TDR	TDR Page
129	Implicit	Physical Page	TDCX	TDCX Page
130	Implicit	Physical Page	TDVPR	TDVPR Page
132	Implicit	Physical Page	REG_PAGE	PT_REG private page
144	Implicit	TD logical control structure	TDCS	TDCS Logical Structure
145	Implicit	TD logical control structure	TDVPS	TDVPS Logical Structure
146	Implicit	TD logical control structure	SEPT_TREE	Secure EPT Tree
147	Implicit	TD logical control structure	SEPT_ENTRY	Secure EPT Entry
168	Implicit	Component of logical control structure	RTMR	TDCS.RTMR
169	Implicit	Component of logical control structure	TD_EPOCH	TDCS.TD_EPOCH
170	Implicit	Component of logical control structure	STDBC	TDCS.STDBC_LINK
171	Implicit	Component of logical control structure	MIGSC	TDCS.MIGSC_LINK and MIGSC page

Operand ID	Explicit/ Implicit	Class	Operand	Description
172	Implicit	Component of logical control structure	OP_STATE	TDCS.OP_STATE
173	Implicit	Component of logical control structure	MIG	TDCS Migration Context
174	Implicit	Component of logical control structure	SERVTD_BINDINGS	Service TD bindings table
184	Implicit	Abstract item	SYS	Intel TDX Module
185	Implicit	Abstract item	TDMR	TDMR
186	Implicit	Abstract item	KOT	KOT
187	Implicit	Abstract item	KET	KET
188	Implicit	Abstract item	WBCACHE	TDH.PHYMEM.CACHE.WB State

4. ABI Reference: Data Types

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

4.1. Basic Crypto Types

Table 4.1: 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
KEY256	32	256-bit key

4.2. UPDATED: TDX Module Configuration, Enumeration and Initialization Types

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

4.2.1. 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 4.2: 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

4.2.2. UPDATED: 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.

TDSYSINFO_STRUCT's size is 1024B.

Table 4.3: 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
	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

Section	Field Name	Offset (Bytes)	Type	Size (Bytes)	Description
	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
	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	

Section	Field Name	Offset (Bytes)	Type	Size (Bytes)	Description
					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 4.2.1 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

4.2.3. UPDATED: 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.

Table 4.4: 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 a MAX_CMRS (32) entry array of CMR_INFO entries. The CMRs are sorted from the lowest base address to the highest base address, and they are non-overlapping.

4.2.4. UPDATED: 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 4.5: 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.

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 4.2.25.1.
- For backward compatibility, this value is also enumerated by TDSYSINFO_STRUCT.MAX_RESREVED_PER_TDMR (see 4.2.2).
- 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.

4.3. TD Parameters Types

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

4.3.1. UPDATED: ATTRIBUTES

ATTRIBUTES is defined as a 64b field that specifies various 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 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 three groups, as shown in the table below. If any bit in the TUD group is set to 1, the guest TD is under off-TD debug and is untrusted. The SEC group indicates features that may impact TD security but are not considered as impacting TD trust.

Table 4.6: ATTRIBUTES Definition

Bits	Group	Description	Bits	Field	Description
7:0	**TUD**	TD Under Debug If any of the bits in this group are set to 1, the guest TD is untrusted.	0	**DEBUG**	Guest TD runs in off-TD debug mode. Its VCPU state and private memory are accessible by the host VMM.
			7:1	**RESERVED**	Reserved for future TUD flags – must be 0
31:8	**SEC**	Attributes that may impact TD security	28:8	**RESERVED**	Reserved for future SEC flags – must be 0
			29	**MIGRATABLE**	TD is migratable (using a Migration TD)
			30	**PKS**	TD is allowed to use Supervisor Protection Keys.
			31	**KL**	TD is allowed to use Key Locker. Must be 0
63:32	**OTHER**	Attributes that do not impact TD security	62:32	**RESERVED**	Reserved for future OTHER flags – must be 0
			63	**PERFMON**	TD is allowed to use Perfmon and PERF_METRICS capabilities.

4.3.2. 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 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 Spec] for details.

4.3.3. 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 4.7: 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

4.3.4. UPDATED: 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/RDM or TDH.SYS.INFO.

TD_PARAMS' size is 1024B.

Table 4.8: UPDATED: TD_PARAMS Definition

Field	Offset (Bytes)	Type	Size (Bytes)	Description	Included in TDREPORT?
ATTRIBUTES	0	64b bitmap (see 4.3.1)	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
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 illegal 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	No
RESERVED	18	N/A	6	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 **Bits 63:6** Reserved – must be 0	No

Field	Offset (Bytes)	Type	Size (Bytes)	Description	Included in TDREPORT?
EXEC_CONTROLS	32	64b bitmap (see Table 4.9)	8	Non-measured TD-scope execution controls	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
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
RESERVED	224	N/A	32	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

Table 4.9: TD_PARAMS_STRUCT.EXEC_CONTROLS Definition

Bits	Name	Description
0	**GPAW**	TD-scope Guest Physical Address Width execution control: copied to each TD VMCS **GPAW** execution control on TDH.VP.INIT 0: GPA.SHARED bit is GPA[47] 1: GPA.SHARED bit is GPA[51]
63:1	**RESERVED**	Must be 0

4.4. 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 Spec].

4.4.1. Physical Page Size

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

Table 4.10: Page Size Definition

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

4.5. UPDATED: 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.

4.5.1. 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 4.11: EPT Levels Definition

Level	0	1	2	3	4	5 (5-Level EPT Only)
GPA Range Size	4KB	2MB	1GB	512GB	256TB	16PB[1]
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
GPA Offset Bits	20:12	29:21	38:30	47:39	51:48 (5-level EPT only)	N/A

4.5.2. 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.

4.5.2.1. *Returned Secure EPT Entry Content*

The returned 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.

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

Table 4.12: Secure EPT Entry Content as Returned by TDX Interface Functions

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	6	0	0
6	6	1	IPAT	Ignore PAT	N/A	1	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	VPW	Verify Paging-Write	No	0	0	0
58	58	1	PW	Paging Write	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

4.5.2.2. *Additional Returned Secure EPT Information*

Additional information for secure EPT entries is returned as defined below.

Table 4.13: Additional Secure EPT Entry Information Returned by TDX Interface Functions

Bits	Name	Description
2:0	Level	Level of the returned Secure EPT entry – see 4.5.1 above
7:3	Reserved	Set to 0
15:8	State	The TDX state of the Secure EPT entry – see Table 4.14 below
63:16	Reserved	Set to 0

Table 4.14: Secure EPT Entry TDX State Returned by TDX Interface Functions

State Name	Public State Number	Description
FREE	0	Secure EPT entry does not map a GPA range.
REMOVED	5	Secure EPT entry is of a removed page

State Name	Public State Number	Description
MAPPED	4	Secure EPT entry maps a private GPA range which is accessible by the guest TD.
BLOCKED	1	Secure EPT entry maps a private GPA range, but new address translations to that range are blocked.
BLOCKEDW	8	Secure EPT entry maps a private GPA range, but new address translations for write operations to that range are blocked.
EXPORTED_BLOCKEDW	9	Secure EPT entry maps a private page that has been blocked for writing and exported.
EXPORTED_DIRTY	11	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	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	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	Secure EPT entry is both pending and blocked.
PENDING_BLOCKEDW	16	Secure EPT entry is both pending and blocked for writing.
PENDING_EXPORTED_BLOCKEDW	17	Secure EPT entry is both pending and exported.
PENDING_EXPORTED_DIRTY	19	Secure EPT entry is both pending and exported, and is not blocked for writing.
PENDING_EXPORTED_DIRTY_BLOCKEDW	20	Secure EPT entry is both pending and exported, and is blocked for writing.

4.6. TD Entry and Exit Types

4.6.1. 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 4.15: Extended Exit Qualification

Bits	Name	Description		
3:0	TYPE	Extended exit qualification type		
		Value	**Name**	**Description**
		0	NONE	No extended exit qualification
		1	ACCEPT	Exit qualification for an EPT violation during TDG.MEM.PAGE.ACCEPT
		Other	Reserved	
31:4	Reserved	Set to 0		
63:32	INFO	TYPE-specific information Set to 0 for NONE – See below for other values of TYPE		

Table 4.16: Extended Exit Qualification INFO Field for ACCEPT

Bits	Name	Description
34:32	REQ_SEPT_LEVEL	SEPT level requested as an input to TDG.MEM.PAGE.ACCEPT
37:35	ERR_SEPT_LEVEL	SEPT level in which TDG.MEM.PAGE.ACCEPT detected an error
45:38	ERR_SEPT_STATE	The TDX state of the Secure EPT entry where TDG.MEM.PAGE.ACCEPT detected an error – see Table 4.14 above
46	ERR_SEPT_IS_LEAF	Indicates that the SEPT entry where TDG.MEM.PAGE.ACCEPT detected an error is a leaf entry
63:47	Reserved	Set to 0

4.7. Measurement and Attestation Types

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

4.7.1. 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.

4.7.2. TDREPORT_STRUCT

TDREPORT_STRUCT is the output of the TDG.MR.REPORT function. TDREPORT_STRUCT is composed of a generic MAC structure (REPORTMACSTRUCT, see 4.7.3 below), a SEAMINFO structure and a TDX-specific TEE info structure (TDINFO_STRUCT, see 4.7.5 below).

The size of TDREPORT_STRUCT is 1024B.

Table 4.17: 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	512	TD's attestable properties

4.7.3. REPORTMACSTRUCT (Reference)

REPORTMACSTRUCT is common to Intel's Trusted Execution Environments (TEEs) – e.g., SGX and TDX. REPORTMACSTRUCT is the first field in the TEE report structure. 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 first confirm that its REPORTMACSTRUCT.TEE_TCB_INFO_HASH equals the hash of the TEE_TCB_INFO_STRUCT (if applicable) and that REPORTMACSTRUCT.TEE_INFO_HASH equals the hash of the TDINFO_STRUCT. Then, software uses ENCLU(EVERIFYREPORT) to help check the integrity of the REPORTMACSTRUCT. If all checks pass, the measurements in the structure describe a TEE on this platform.

The size of REPORTMACSTRUCT is 256B.

Table 4.18: 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
CPUSVN	16	CPUSVN	16	CPU SVN	Yes
TEE_TCB_INFO_HASH	32	SHA384_HASH	48	SHA384 of TEE_TCB_INFO for TEEs implemented using Intel TDX	Yes
TEE_INFO_HASH	80	SHA384_HASH	48	SHA384 of TEE_INFO: a TEE-specific info structure (TDG.VP.INFO 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

4.7.4. REPORTTYPE (Reference)

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

The size of REPORTTYPE is 4B.

Table 4.19: REPORTTYPE Definition

Name	Offset (Bytes)	Size (Bytes)	Description
TYPE	0	1	Trusted Execution Environment (TEE) Type: 0x00: SGX 0x7F-0x01: Reserved (TEE implemented by CPU) 0x80: Reserved (TEE implemented by SEAM module) 0x81: TDX 0xFF-0x82: Reserved (TEE implemented by SEAM module)
SUBTYPE	1	1	TYPE-specific subtype Value is 0
VERSION	2	1	TYPE-specific version. Value is 0
RESERVED	3	1	Must be zero

4.7.5. UPDATED: 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 Spec] for additional details.

The size of TDINFO_STRUCT is 512B.

Table 4.20: TDINFO_STRUCT 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	NUM_RTMRS * 48	Array of NUM_RTMRS (4) run-time extendable measurement registers
SERVTD_HASH	400	SHA384_HASH	48	Hash of the TDINFO_STRUCTs of service TDs bound or pre-bound to this TD

Name	Offset (Bytes)	Type	Size (Bytes)	Description
RESERVED	448	N/A	64	Must be zero

4.8. UPDATED: 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 Spec].

4.8.1. MD_FIELD_ID: Metadata Field Identifier and Sequence Header

A metadata field identifier is used for accessing a single field. A metadata sequence header is an extension of the field identifier, used for accessing a field sequence.

Lists of metadata field identifiers for global-scope metadata, TD-scope metadata and VCPU-scope metadata are provided in Ch. 5. To access a certain metadata field, a **base identifier** is taken from those tables; only LAST_FIELD_IN_SEQUENCE and WRITE_MASK_VALID need to be updated as needed.

Table 4.21: Metadata Field Identifier and Sequence Header Definition

Bits	Size	Name	Base Identifier from Metadata Tables	Description
31:0	32	**FIELD_CODE**	Yes	For a single metadata field identifier, identifies the element that is being accessed. For a metadata sequence header, identifies the first field that is being accessed in a sequence.
33:32	2	**ELEMENT_SIZE_CODE**	Yes	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**	Yes	Number of elements in a metadata field, minus 1 For a single-element identifier, LAST_ELEMENT_IN_FIELD is ignored.
46:38	9	**LAST_FIELD_IN_SEQUENCE**	No	Number of fields in a sequence, minus 1 For a single field identifier, LAST_FIELD_IN_SEQUENCE is 0.
49:47	3	**RESERVED**	Yes	Must be 0

Bits	Size	Name	Base Identifier from Metadata Tables	Description
50	1	**INC_SIZE**	Yes	For a single field identifier, INC_SIZE must be 0. 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	Indicates that a write mask is provided together with the write value. Applicable only when used for writing fields. For backward compatibility, TDH.MNG.WR and TDH.VP.WR version 0 ignore this field and use a default value of 1.
54:52	3	**CONTEXT_CODE**	Yes	Specifies the context of the field: **0:** Platform (whole Intel TDX module) **1:** TD **2:** TD VCPU **Other:** Reserved For backward compatibility, TDH.MNG.RD, TDH.MNG.WR, TDH.VP.RD and TDH.VP.WR version 0 ignore this field and use an implicit value.
55	1	**RESERVED**	Yes	Must be 0
61:56	6	**CLASS_CODE**	Yes	Identifies the class of the fields being accessed Class codes are defined in 4.8.3.
62	1	**RESERVED**	Yes	Must be 0
63	1	**NON_ARCH**	Yes	Specified 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.

4.8.2. 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 4.22: Meaning of Field Codes

Field Class	Field Code Meaning	Reference
VMCS	Field code is identical to 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]
MSR Bitmap	Offset (in 8B units) from the beginning of the architectural MSR bitmaps page structure Offset (in 8B units) from the beginning of the architectural MSR bitmaps page structure	[Intel SDM, Vol. 3, 24.6.9]
Secure EPT Root	Offset (in 8B units) from the beginning of the page	

Field Class	Field Code Meaning		Reference
Virtual APIC Page	Offset (in 8B units) from the beginning of the architectural virtual APIC page structure		[Intel SDM, Vol. 3, 29.1]
CPUID Config	CPUID Leaf and Sub-Leaf, packed as shown below:		
	Bits	**Description**	
	31	Leaf number bit 31	
	30:24	Leaf number bit 6:0	
	23	Sub-leaf not applicable flag	
	22:16	Sub-leaf number bits 6:0	
	15:1	Reserved, must be 0	
	0	Element index within field	
GPR State	Architectural GPR number		
MSR State	Architectural MSR index, packed as shown below:		
	Bits	**Description**	
	31:14	Reserved, must be 0	
	13	Bit 31 (equal to bit 30) of the architectural MSR index	
	12:0	Bits 12:0 of the architectural MSR index	
Extended State	Offset (in 8B units) from the beginning of the page extended state buffer		
Other	Arbitrary field identifiers		

4.8.3. Class Codes

4.8.3.1. *NEW: TDX Module Global Scope Field Class Codes*

TDX Module global scope field classes are defined as follows:

Table 4.23: 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
24	TD Control Structures
25	TD Configurability
32	Migration
33	Service TD

4.8.3.2. *UPDATED: TD-Scope (TDR and TDCS) Field Class Codes*

TD-scope field classes are defined as follows:

Table 4.24: TD Scope (TDR and TDCS) Field Class Codes Definition

Class Code	Control Structure	Field Class Name
0	TDR	TD Management
1	TDR	Key Management
16	TDCS	TD Management
17	TDCS	Execution Controls
18	TDCS	TLB Epoch Tracking
19	TDCS	Measurement
20	TDCS	CPUID
24	TDCS	Migration
25	TDCS	Service TD
32	TDCS	MSR Bitmaps
33	TDCS	Secure EPT Root
35	TDCS	MIGSC Links

4.8.3.3. *VCPU-Scope (TDVPS) Field Class Codes*

TDVPS field classes are defined as follows:

Table 4.25: 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	Other Guest State
18	Guest Extended State
19	Guest MSR State
32	Management

4.8.4. 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

4.8.5. MD_LIST_HEADER: Metadata List Header

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

Table 4.26: 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

4.8.6. 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 4.27: 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

4.8.7. 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 4.28: 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

4.8.8. 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 4.29: 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

4.9. NEW: Service TD Types

4.9.1. SERVTD_TYPE: Service TD Binding Type

SERVTD_TYPE is a 16-bit field which specifies the binding type of a service TD.

Table 4.30: SERVTD_TYPE Definition

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

4.9.2. SERVTD_ATTR: Service TD Binging Attributes

SERVTD_ATTR is a 64-bit field which specifies binding attributes of a service TD.

Table 4.31: SERVTD_ATTR Definition

Bit(s)	Name	Description
0	INSTANCE_BINDING	0: **Class Binding:** Rebinding can be done with any TD with the same SERVTD_INFO_HASH, SERVTD_TYPE and SERVTD_ATTR as the original binding. Those parameters are migrated. 1: **Instance Binding:** Rebinding can be done with the same TD instance (same SERVTD_UUID), using the same SERVTD_TYPE and SERVTD_ATTR as the original binding.
1	RESERVED	Must be 0
2	RESERVED	Must be 0
3	RESERVED	Must be 0
31:4	RESERVED	Must be 0
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 MROWNER 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

Bit(s)	Name	Description
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
63:43	RESERVED	Must be 0

4.10. NEW: Migration Types

4.10.1. MBMD: Migration Bundle Metadata

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

4.10.1.1. *Generic MBMD Structure*

Table 4.32: 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
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

Field	Offset (Bytes)	Size (Bytes)	Description	Included In MAC	Included in IV
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

4.10.1.2. *TD-Scope Immutable Non-Memory State MBMD Fields*

Table 4.33: TD-Scope Immutable Non-Memory State MBMD Fields Definition

Field	Offset (Bytes)	Size (Bytes)	Description	Included In MAC	Included in IV
NUM_MIGS	24	2	Maximum number of migration streams that will be used	Yes	No
NUM_IN_ORDER_MIGS	26	2	Number of migration streams that will be used during the in-order migration phase	Yes	No
RESERVED	28	4	Maximum number of migration streams that will be used	Yes	No

4.10.1.3. *TD-Scope Mutable Non-Memory State MBMD Fields*

Table 4.34: 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

4.10.1.4. *VCPU-Scope Mutable Non-Memory State MBMD Fields*

Table 4.35: 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

4.10.1.5. *TD Private Memory MBMD Fields*

Table 4.36: 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, including PENDING pages and CANCEL requests for which no content is migrated	Yes	No
RESERVED	26	6	Reserved, must be 0	Yes	No

4.10.1.6. *Epoch Token MBMD Fields*

Table 4.37: 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

4.10.1.7. *Abort Token MBMD Fields*

Table 4.38: 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

4.10.2. 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).

4.10.2.1. *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 4.39: GPA_LIST_INFO

Bits	Name	Description
2:0	**RESERVED**	Reserved: must be 0
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
63:55	**LAST_ENTRY**	Index of the last entry in the GPA list

4.10.2.2. *GPA List Entry*

Table 4.40 below shows the format of a GPA list entry as used. Bits that are located in the same place as in SEPT entries are highlighted. 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 4.40: 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
0	1	R	Read	Ignored	Unmod.	Ignored	Yes	Yes	Unmod.	Ignored	Unmod.

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	1	W	Write	Ignored	Unmod.	Ignored	Yes	Yes	Unmod.	Ignored	Unmod.
2	1	Xs	Execute (Supervisor)	Ignored	Unmod.	Ignored	Yes	Yes	Unmod.	Ignored	Unmod.
4:3	2	OPERATION	See below	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
6:5	2	RESERVED	Reserved	Must be 0	Unmod.	Must be 0	Unmod.	Must be 0	Unmod.	Must be 0	Unmod.
9:7	3	MIG_TYPE	See below	Yes	Unmod.	Yes	Unmod.	Yes	Unmod.	Yes	Unmod.
10	1	Xu	Execute (User)	Ignored	Unmod.	Ignored	Yes	Yes	Unmod.	Ignored	Unmod.
11	1	PENDING	See below	Ignored	Unmod.	Ignored	Yes	Yes	Unmod.	Ignored	Unmod.
51:12	40	GPA	Guest Physical Address bits 51:12	Yes	Unmod.	Yes	Unmod.	Yes	Unmod.	Yes	Unmod.
56:52	5	STATUS	See below	Must be 0	Yes	Must be 0	Yes	Must be 0	Yes	Must be 0	Yes
57	1	VPW	Verify Paging-Write	Ignored	Unmod.	Ignored	Yes	Yes	Unmod.	Ignored	Unmod.
58	1	PW	Paging Write	Ignored	Unmod.	Ignored	Yes	Yes	Unmod.	Ignored	Unmod.
59	1	RESERVED	Reserved	Must be 0	Unmod.	Must be 0	Unmod.	Must be 0	Unmod.	Must be 0	Unmod.
60	1	SSS	Supervisor Shadow Stack	Ignored	Unmod.	Ignored	Yes	Yes	Unmod.	Ignored	Unmod.
62:61	2	RESERVED	Reserved	Must be 0	Unmod.	Must be 0	Unmod.	Must be 0	Unmod.	Must be 0	Unmod.
63	1	SVE	Suppress #VE	Ignored	Unmod.	Ignored	Yes	Yes	Unmod.	Ignored	Unmod.

4.10.2.3. *GPA List Entry Details*

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 4.41: OPERATION Values Definition for TDH.EXPORT.BLOCKW

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

Table 4.42: OPERATION Values Definition for TDH.EXPORT.MEM

Value	Name	Input	Output
0	NOP	No operation	Not exported
1	MIGRATE	Export	Initial export during this migration session or following a CANCEL
2	CANCEL	Cancel previous export	Cancellation of a previous migration
3	RE_MIGRATE	Export	Re-export of updated content or attributes

Table 4.43: OPERATION Values Definition for TDH.IMPORT.MEM

Value	Name	Input	Output
0	NOP	No operation	Not imported
1	MIGRATE	Initial import during this migration session or following a CANCEL	Imported
2	CANCEL	Cancel previous import	Removed previous import
3	RE_MIGRATE	Re-export of updated page content or attributes	Imported

Table 4.44: OPERATION Values Definition for TDH.EXPORT.RESTORE

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

GPA List Details: MIG_TYPE

Table 4.45: MIG_TYPE Values Definition

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

GPA List Details: PENDING

Table 4.46: PENDING Values Definition

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

GPA List Details: STATUS

Table 4.47: 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	Secure EPT entry was busy
4	SEPT_ENTRY_STATE_INCORRECT	Secure EPT entry state was incorrect for the requested operation and the TD's OP_STATE

Value	Name	Description
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	TD page was busy
31-13	Reserved	Reserved

4.10.3. 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.

4.10.3.1. *Migration Buffers List Entry*

Table 4.48: 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

4.10.4. Memory Migration Page MAC List

A page MAC list specifies a list of MACs over 4KB migrated pages and their GPA 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.

4.10.5. 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.

4.10.5.1. *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 4.49: 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 GPA list page, which must be a shared HPA
54:52	**RESERVED**	Reserved: must be 0
63:55	**LAST_ENTRY**	Index of the last entry in the page list

5. UPDATED: ABI Reference: Metadata (Non-Memory State)

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

5.1. NEW: 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.

5.1.1. How to Read the Global Fields Table

The access columns describe whether this field is accessible to the host VMM and to guest TDs. Access is done using the TDX module metadata access functions, e.g., TDH.SYS.RD. Possible values are shown in the table below.

Table 5.1: Field Access Definition

Access	Meaning
None	No access to the field
RO	This field can only be read.

5.1.2. Global Metadata Fields

Table 5.2: Global Scope Metadata

Class	Field	VMM Access	Guest Access	Type	Field Size (Bytes)	Num Fields	Num Elem.	Elem. Size (Bytes)	Description	Base FIELD_ID (Hex)
Platform Info	NUM_PKGS	RO	None	Integer	4	1	1	4	Number of CPU packages in the system (1 - 8)	0x0000000200000000
Platform Info	PKG_FMS	RO	None	N/A	4	8	1	4	Array of version information (type, family, model, stepping) as returned by CPUID(1).EAX for each package. Unused entries (NUM_PKGS and above) are set to 0.	0x0000000200000001
TDX Module Version	VENDOR_ID	RO	RO	Integer	4	1	1	4	0x8086 for Intel	0x0800000200000000
TDX Module Version	BUILD_DATE	RO	None	BCD	4	1	1	4	Intel TDX module build data – in yyyymmdd BCD format (each digit occupies 4 bits)	0x8800000200000001
TDX Module Version	BUILD_NUM	RO	None	Integer	2	1	1	2	Build number of the Intel TDX module	0x8800000100000002
TDX Module Version	MINOR_VERSION	RO	RO	Integer	2	1	1	2	Minor version number of the Intel TDX module	0x0800000100000003
TDX Module Version	MAJOR_VERSION	RO	RO	Integer	2	1	1	2	Major version number of the Intel TDX module	0x0800000100000004
TDX Module Info	SYS_ATTRIBUTES	RO	RO	Bitmap	4	1	1	4	Module attributes Bits 30:0 Reserved – set to 0 Bit 31 0 indicates a production module. 1 indicates a debug module.	0x0A00000200000000
TDX Module Info	NUM_TDX_FEATURES	RO	RO	8-bit integer	1	1	1	1	Number of TDX_FEATURES fields	0x0A00000000000001
TDX Module Info	TDX_FEATURES0	RO	RO	64-bit bitmap	8	1	1	8	Enumerates TDX features: Bit 0 TD Migration Bit 1 TD Preserving Bit 2 Service TD Bit 3 TDG.VP.RD/WR Bit 4 Relaxed mem management concurrency Bits 63:5 Reserved, set to 0	0x0A00000300000008
CMR Info	NUM_CMRS	RO	None	Integer	2	1	1	2	Number of the following CMR entries	0x9000000100000000
CMR Info	CMR_BASE	RO	None	Physical Address	8	32	1	8	Array of CMR base addresses Since a CMR is aligned on 4KB, bits 11:0 are 0.	0x9000000300000080
CMR Info	CMR_SIZE	RO	None	Integer	8	32	1	8	Array of CMR sizes, in bytes Since a CMR is aligned on 4KB, bits 11:0 are 0. A value of 0 indicates a null entry.	0x9000000300000100
TDMR Info	MAX_TDMRS	RO	None	Integer	2	1	1	2	The maximum number of TDMRs supported	0x9100000100000008
TDMR Info	MAX_RESERVED_PER_TDMR	RO	None	Integer	2	1	1	2	The maximum number of reserved areas per TDMR	0x9100000100000009

Class	Field	VMM Access	Guest Access	Type	Field Size (Bytes)	Num Fields	Num Elem.	Elem. Size (Bytes)	Description	Base FIELD_ID (Hex)
TDMR Info	PAMT_4K_ENTRY_SIZE	RO	None	Integer	2	1	1	2	The size of a PAMT_4K (1 entry per 4KB of TDMR) entry, in bytes – determines the number of bytes that need to be reserved for the PAMT_4K area.	0x9100000100000010
TDMR Info	PAMT_2M_ENTRY_SIZE	RO	None	Integer	2	1	1	2	The size of a PAMT_2M (1 entry per 2MB of TDMR) entry, in bytes – determines the number of bytes that need to be reserved for the PAMT_2M area.	0x9100000100000011
TDMR Info	PAMT_1G_ENTRY_SIZE	RO	None	Integer	2	1	1	2	The size of a PAMT_1G (1 entry per 1GB of TDMR) entry, in bytes – determines the number of bytes that need to be reserved for the PAMT_1G area.	0x9100000100000012
TD Control Structures	TDR_BASE_SIZE	RO	None	Integer	2	1	1	2	Base value for the number of bytes required to hold TDR	0x9800000100000000
TD Control Structures	TDCS_BASE_SIZE	RO	None	Integer	2	1	1	2	Base value for the number of bytes required to hold TDCS	0x9800000100000100
TD Control Structures	TDVPS_BASE_SIZE	RO	None	Integer	2	1	1	2	Base value for the number of bytes required to hold TDVPS	0x9800000100000200
TD Configurability	ATTRIBUTES_FIXED0	RO	None	Bitmap	8	1	1	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.	0x1900000300000000
TD Configurability	ATTRIBUTES_FIXED1	RO	None	Bitmap	8	1	1	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.	0x1900000300000001
TD Configurability	XFAM_FIXED0	RO	None	Bitmap	8	1	1	8	If any certain bit is 0 in XFAM_FIXED0, it must be 0 in any TD's XFAM.	0x1900000300000002
TD Configurability	XFAM_FIXED1	RO	None	Bitmap	8	1	1	8	If any certain bit is 1 in XFAM_FIXED1, it must be 1 in any TD's XFAM.	0x1900000300000003
TD Configurability	NUM_CPUID_CONFIG	RO	None	Unsigned 16-bit integer	2	1	1	2	Number of the following CPUID_CONFIG entries	0x9900000100000004
TD Configurability	CPUID_CONFIG_LEAVES	RO	None		8	40	1	8	Array of CPUID leaf / sub-leaf numbers: Bits 31:0: Leaf number Bits 63:32 Sub-leaf number. A value of -1 indicates a CPUID leaf with no sub-leaves.	0x9900000300000400
TD Configurability	CPUID_CONFIG_VALUES	RO	None		16	40	2	8	Array of configurable virtualization of the value returned by CPUID: Element 0[31:0]: EAX Element 0[63:32]: EBX Element 1[31:0]: ECX Element 1[63:32]: EDX A bit value of 1 indicates that the host VMM can configure that bit	0x9900000300000500
Migration	MIG_ATTRIBUTES	RO	RO	64-bit bitmap	8	1	1	8	Migration attributes (details are TBD)	0xA000000300000000
Migration	MIN_EXPORT_VERSION	RO	RO	16-bit integer	2	1	1	2	Minimum value of migration version supported for export	0x2000000100000001
Migration	MAX_EXPORT_VERSION	RO	RO	16-bit integer	2	1	1	2	Maximum value of migration version supported for export	0x2000000100000002
Migration	MIN_IMPORT_VERSION	RO	RO	16-bit integer	2	1	1	2	Minimum value of migration version supported for import	0x2000000100000003
Migration	MAX_IMPORT_VERSION	RO	RO	16-bit integer	2	1	1	2	Maximum value of migration version supported for import	0x2000000100000004
Migration	MAX_MIGS	RO	None	Unsigned integer	2	1	1	2	Maximum number of migration streams per TD	0xA000000100000010
Migration	NUM_IMMUTABLE_STATE_PAGES	RO	None	Integer	1	1	1	1	Number of pages required for exporting immutable state by TDH.EXPORT.STATE.IMMUTABLE	0xA000000000000020
Migration	NUM_TD_STATE_PAGES	RO	None	Integer	1	1	1	1	Number of pages required for exporting TD state by TDH.EXPORT.STATE.TD	0xA000000000000021
Migration	NUM_VP_STATE_PAGES	RO	None	Integer	1	1	1	1	Number of pages required for exporting VCPU state by TDH.EXPORT.STATE.VP	0xA000000000000022
Service TD	MAX_SERV_TDS	RO	None	Unsigned integer	2	1	1	2	Maximum number of service TDs per TD	0xA100000100000000

Class	Field	VMM Access	Guest Access	Type	Field Size (Bytes)	Num Fields	Num Elem.	Elem. Size (Bytes)	Description	Base FIELD_ID (Hex)
Service TD	SERVTD_ATTR_FIXED0	RO	None	64-bit bitmap	8	1	1	8	Fixed-0 bits of Service TD attributes. A bit value of 0 indicates corresponding SERVTD_ATTR bit must be 0	0xA100000300000001
Service TD	SERVTD_ATTR_FIXED1	RO	None	64-bit bitmap	8	1	1	8	Fixed-1 bits of Service TD attributes. A bit value of 1 indicates corresponding SERVTD_ATTR bit must be 1	0xA100000300000002

5.2. UPDATED: TD-Scope Metadata

TD-scope control structures are described the [TDX Module Spec].

5.2.1. UPDATED: How to Read the TDR and TDCS Tables

The access columns describe whether this field is accessible to the host VMM in production mode (ATTRIBUTES.DEBUG == 0) and debug mode (ATTRIBUTES.DEBUG == 1), to the guest TD and to a Migration TD. Access is done using the TDX module metadata access functions, e.g., TDH.MNG.RD. Possible values are shown in the table below.

Table 5.3: Field Access Definition

Access	Meaning
None	No access to the field
RO	This field can only be read.
RW	This field can be read and written.
RWS	This field can be read and written. The TDX module performs some special operation on write.

5.2.2. UPDATED: 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 5.4: 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 Spec] for details.

Note: The table below lists only TDR fields that may be accessed by the host VMM in either production or debug mode.

Table 5.5: TDR Definition

Class	Field	VMM Prod. Access	VMM Debug Access	Guest Access	MigTD Access	Type	Field Size (Bytes)	Num Fields	Num Elem.	Elem. Size (Bytes)	Description	Base FIELD_ID (Hex)
TD Management	FATAL	RO	RO	None	None	Boolean	Non-Arch	1	1	Non-Arch	Indicates a fatal error, e.g., #MC during TD operation.	0x8010000300000001
TD Management	NUM_TDCX	RO	RO	None	None	Unsigned Integer	Non-Arch	1	1	Non-Arch	Number of TDCX pages that have been added by TDH.MNG.ADDCX	0x8010000300000002

Class	Field	VMM Prod. Access	VMM Debug Access	Guest Access	MigTD Access	Type	Field Size (Bytes)	Num Fields	Num Elem.	Elem. Size (Bytes)	Description	Base FIELD_ID (Hex)
TD Management	CHLDCNT	RO	RO	None	None	64b Unsigned Integer	8	1	1	8	The number of 4KB child pages (including opaque control structure pages) associated with this TDR	0x8010000300000004
TD Management	LIFECYCLE_STATE	RO	RO	None	None	LIFECYCLE_STATE	Non-Arch	1	1	Non-Arch	The life cycle state of this TD	0x8010000300000005
TD Management	TDCX_PA	RO	RO	None	None	Array of Physical Address	8	5	1	8	Physical addresses of the TDCX pages (without HKID bits)	0x8010000300000010
TD Management	TD_UUID	RO	RO	RO	RO	256-bit blob	32	1	4	8	Universally Unique Identifier of the TD	0x8010000300000020
Key Management	HKID	RO	RO	None	None	16b Unsigned Integer	2	1	1	2	Private HKID	0x8110000100000001
Key Management	PKG_CONFIG_BITMAP	RO	RO	None	None	Bitmap	Non-Arch	1	1	Non-Arch	Bitmap that indicates on which package TDH.MNG.KEY.CONFIG was executed successfully using this private key entry	0x8110000300000002

5.2.3. UPDATED: 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 5.6: 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 Spec] for details
Measurement	TD measurement registers and associated fields – see the [TDX Module Spec] for details
Migration	TDCS fields that control TD migration
MIGSC Links	Links to Migration Stream Context pages
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

Note: The table below lists only TDCS fields that may be accessed by the host VMM in either production or debug mode.

Table 5.7: TDCS Definition

Class	Field	VMM Prod. Access	VMM Debug Access	Guest Access	MigTD Access	Type	Field Size (Bytes)	Num Fields	Num Elem.	Elem. Size (Bytes)	Description	Base FIELD_ID (Hex)
TD Management	NUM_VCPUS	RO	RO	RO	None	32b Unsigned Integer	4	1	1	4	The number of VCPUs that are either in TDX non-root mode (TDVPS.VCPU_STATE == VCPU_ACTIVE) or are ready to run (TDVPS.VCPU_STATE == VCPU_READY):	0x9010000200000001

Class	Field	VMM Prod. Access	VMM Debug Access	Guest Access	MigTD Access	Type	Field Size (Bytes)	Num Fields	Num Elem.	Elem. Size (Bytes)	Description	Base FIELD_ID (Hex)
											this includes VCPUs that have been successfully initialized (by TDH.VP.INIT) or imported (by TDH.IMPORT.STATE.VP) and have not since started teardown (due to a Triple Fault)	
TD Management	NUM_ASSOC_VCPUS	RO	RO	None	None	32b Unsigned Integer	4	1	1	4	The number of VCPUS associated with LPs – i.e., the LPs might hold TLB translations and/or cached TD VMCS	0x9010000200000002
TD Management	OP_STATE	RO	RO	None	None	OP_STATE	Non-Arch	1	1	Non-Arch	The operation state (sub-state of life cycle TD_KEYS_CONFIGURED state) of this TD	0x9010000300000004
Execution Controls	ATTRIBUTES	RO	RO	RO	None	ATTRIBUTES	8	1	1	8	TD attributes	0x1110000300000000
Execution Controls	XFAM	RO	RO	RO	None	XCR0	8	1	1	8	Extended Features Available Mask: indicates the extended user and system features which are available for the TD. Copied to each TDVPS on TDH.VP.INIT.	0x1110000300000001
Execution Controls	MAX_VCPUS	RO	RO	RO	None	32b Unsigned Integer	4	1	1	4	Maximum number of VCPUs	0x1110000200000002
Execution Controls	GPAW	RO	RO	RO	None	Boolean	Non-Arch	1	1	Non-Arch	This bit has the same meaning as the VMCS GPAW execution control: 0: GPA.SHARED bit is GPA[47] 1: GPA.SHARED bit is GPA[51]	0x1110000300000003
Execution Controls	EPTP	RO	RO	None	None	EPTP	8	1	1	8	TD-scope Secure EPT pointer: format is the same as the VMCS EPTP execution control; copied to each TD VMCS EPTP on TDH.VP.INIT	0x1110000300000004
Execution Controls	TSC_OFFSET	RO	RO	None	None	64b unsigned Integer	8	1	1	8	TD-scope TSC offset execution control: copied to each TD VMCS TSC-offset execution control on TDH.VP.INIT	0x111000030000000A
Execution Controls	TSC_MULTIPLIER	RO	RO	None	None	64b Unsigned Integer	8	1	1	8	TD-scope TSC multiplier execution control: copied to each TD VMCS TSC-multiplier execution control on TDH.VP.INIT	0x111000030000000B
Execution Controls	TSC_FREQUENCY	RO	RO	RO	None	16b Unsigned Integer	2	1	1	2	Virtual TSC frequency – in units of 25MHz	0x111000010000000C
Execution Controls	VIRTUAL_TSC	None	None	None	None	64b Unsigned Integer	8	1	1	8	Virtual TSC value. This value is only calculated and used at the end of migration (TDH.EXPORT.STATE.TD) and doesn't need to actually reside in TDCS.	0x111000030000000D
Execution Controls	NUM_CPUID_VALUES	RO	RO	None	None	16b Unsigned Integer	2	1	1	2	Number of valid fields in CPUID_VALUES	0x911000010000000E
Execution Controls	XBUFF_SIZE	RO	RO	None	None	Unsigned Integer	2	1	1	2	Actual size of the XSAVE buffer – calculated by TDH.MNG.INIT based on XFAM	0x911000010000000F
Execution Controls	NOTIFY_ENABLES	None	RW	RW	None	Bitmap	8	1	1	8	Enable guest notification of events: bit 0: Notify when Zero Step attack is suspected bits 63:1: Reserved, must be 0	0x9110000300000010
Execution Controls	XBUFF_OFFSETS	RO	RO	None	None	Unsigned Integer	4	19	1	4	XSAVE buffer components offsets – calculated by TDH.MNG.INIT based on XFAM	0x9110000200000800
TLB Epoch Tracking	TD_EPOCH	RO	RO	None	None	64b Integer	8	1	1	8	The TD epoch counter: incremented by the host VMM using the TDH.MEM.TRACK function	0x9210000300000000
TLB Epoch Tracking	REFCOUNT	RO	RO	None	None	16b Unsigned Integer	2	2	1	2	Each REFCOUNT counts the number of LPs which may have TLB entries created during a specific TD_EPOCH and are currently executing in TDX non-root mode.	0x9210000100000001
Measurement	MRTD	RO	RO	RO	None	SHA384_HASH	48	1	6	8	Measurement of the initial contents of the TD	0x1310000300000000
Measurement	MRCONFIGID	RO	RO	RO	None	SHA384_HASH	48	1	6	8	Software-defined ID for non-owner-defined configuration of the guest TD – e.g., run-time or OS configuration	0x1310000300000010
Measurement	MROWNER	RO	RO	RO	None	SHA384_HASH	48	1	6	8	Software-defined ID for the guest TD's owner	0x1310000300000018
Measurement	MROWNERCONFIG	RO	RO	RO	None	SHA384_HASH	48	1	6	8	Software-defined ID for owner-defined configuration of the guest TD – e.g., specific to the workload rather than the run-time or OS	0x1310000300000020
Measurement	RTMR	None	RO	RO	None	Array of SHA384_HASH	48	4	6	8	Array of NUM_RTMRS run-time extendable measurement registers	0x1310000300000040
Measurement	MRTD_CONTEXT	None	None	None	None	SHA384_CONTEXT	Non-Arch	1	Non-Arch	Non-Arch	Non-architectural context used during ongoing calculation of MRTD until TDH.MR.FINALIZE	0x9310000300000080
CPUID	CPUID_VALUES	RO	RO	None	None	CPUID_RET	16	128	2	8	Values returned by CPUID leaves/sub-leaves: Element 0[31:0]: EAX Element 0[63:32]: EBX Element 1[31:0]: ECX Element 1[63:32]: EDX Field code is composed as follows: 31 Leaf number bit 31 30:24 Leaf number bit 6:0 23 Sub-leaf not applicable flag 22:16 Sub-leaf number bits 6:0	0x9410000300000000

Class	Field	VMM Prod. Access	VMM Debug Access	Guest Access	MigTD Access	Type	Field Size (Bytes)	Num Fields	Num Elem.	Elem. Size (Bytes)	Description	Base FIELD_ID (Hex)
											15:1 Reserved, must be 0 0 Element index within field	
Migration	MIG_DEC_KEY_SET	RO	RO	None	None	Boolean	Non-Arch	1	1	Non-Arch	Set when a new MIG_DEC_KEY is written, cleared when the MIG_DEC_KEY is copied to MIG_DEC_WORKING_KEY	0x9810000300000001
Migration	EXPORT_COUNT	RO	RO	None	None	32b Unsigned Integer	4	1	1	4	Counts the number of times this TD has been exported, included aborted export sessions. Incremented at the beginning of each export session (TDH.EPORT.STATE.IMMUTABLE).	0x9810000200000002
Migration	IMPORT_COUNT	RO	RO	None	None	32b Unsigned Integer	4	1	1	4	Counts the number of times this TD has been imported. Incremented by TDH.IMPORT.COMMIT.	0x9810000200000003
Migration	MIG_EPOCH	RO	RO	None	None	32b Unsigned Integer	4	1	1	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.	0x9810000200000004
Migration	BW_EPOCH	RO	RO	None	None	64b Unsigned Integer	8	1	1	8	Blocking-for-write epoch Holds the value of TD_EPOCH at last time TDH.EXPORT.BLOCKW blocked a page for writing.	0x9810000300000005
Migration	TOTAL_MB_COUNT	RO	RO	None	None	Unsigned Integer	8	1	1	8	The total number of migration bundles exported or imported during the current migration sessions	0x9810000300000006
Migration	MIG_DEC_KEY	None	RO	None	RWS	KEY_256	32	1	4	8	Migration decryption key, as written by the Migration TD Special write behavior: - Acquire a shared lock on TDCS.OP_STATE to prevent concurrent migration session start. - Set MIG_DEC_KEY_SET	0x9810000300000010
Migration	MIG_DEC_WORKING_KEY	None	RO	None	RW	KEY_256	32	1	4	8	Migration working key, copied from MIG_KEY at the beginning of a migration session and used throughout the session.	0x9810000300000014
Migration	MIG_VERSION	RO	RO	None	RWS	64b Unsigned Integer	2	1	1	2	Migration protocol version, as written by the migration TD	0x9810000100000020
Migration	MIG_WORKING_VERSION	RO	RO	None	None	16b Unsigned Integer	2	1	1	2	Migration working protocol version, copied from MIG_VERSION at the beginning of a migration session and used throughout the session	0x9810000100000021
Migration	DIRTY_COUNT	RO	RO	None	None	16b Unsigned Integer	8	1	1	8	Counts of the number of pages that must be re-exported, because their contents have been modified since they have been exported, before a start token may be generated	0x9810000300000030
Migration	MIG_COUNT	RO	RO	None	None	64b Unsigned Integer	8	1	1	8	Counts the number of SEPT entries that need to be cleaned up after an aborted migration	0x9810000300000031
Migration	NUM_MIGS	RO	RO	None	None	16b Unsigned Integer	2	1	1	2	Number of Migration Stream Context (MIGSC) pages that have been allocated	0x9810000100000032
Migration	NUM_IN_ORDER_MIGS	RO	RO	None	None	16b Unsigned Integer	2	1	1	2	Number of migration streams that can be used during the export in-order phase	0x9810000100000033
Migration	PRE_IMPORT_UUID	RO	RO	RO	RO	256-bit blob	32	1	4	8	The original value of TD_UUID before is was overwritten as part of the immutable state import	0x9810000300000040
Service TD	SERVTD_HASH	RO	RO	RO	None	SHA384_HASH	48	1	6	8	SHA384 hash of the bound or pre-bound service TDs	0x9910000300000000
Service TD	SERVTD_BINDING_STATE	RO	RO	RO	None	8b Unsigned Integer	1	64	1	1	Indicates the state of the service TD binding slot: 0: NULL 1: PRE_BOUND 2: BOUND 3: UNBOUND	0x9910000000010000
Service TD	SERVTD_INFO_HASH	RO	RO	RO	None	SHA384_HASH	48	64	6	8	SHA384 hash of the service TD's TDINFO_STRUCT	0x9910000300020000
Service TD	SERVTD_TYPE	RO	RO	RO	None	16b Unsigned Integer	2	64	1	2	Service TD type	0x9910000100030000
Service TD	SERVTD_ATTR	RO	RO	RO	None	64b bitmap	8	64	1	8	Service TD attributes	0x9910000300040000
Service TD	SERVTD_UUID	RO	RO	RO	None	256-bit blob	32	64	4	8	Service TD UUID	0x9910000300050000
MSR Bitmaps	MSR_BITMAPS	None	RO	None	None	64b bitmap	8	512	1	8	TD-scope RDMSR/WRMSR exit control bitmaps	0x2010000300000000
Secure EPT Root	SEPT_ROOT	None	RO	None	None	Secure EPT Entry	8	512	1	8	Secure EPT root page (PML5 or PML4)	0x2110000300000000
MIGSC Links	MIGSC_LINKS	RO	RO	None	None	MIGSC_LINK	8	512	1	8	An array of links to Migration Stream Contexts. Each entry contains the following information: Bit 51:12: MIGSC_HPA: Bits 52:12 of the MIGSC page HPA (without the HKID bits) Bit 0: LOCK: Mutex for controlling access to the MIGSC	0xA310000300000000

Class	Field	VMM Prod. Access	VMM Debug Access	Guest Access	MigTD Access	Type	Field Size (Bytes)	Num Fields	Num Elem.	Elem. Size (Bytes)	Description	Base FIELD_ID (Hex)
											Bit 1: INITIALIZED: A boolean flag, indicating that the MIGSC has been initialized. Bit 2: ENABLED: A boolean flag, indicating that the MIGS is enabled The flags are held here, not in the MIGSC itself, to enable efficient state-related operations on all migration streams, e.g., disabling all streams.	

5.3. UPDATED: TDVPS: VCPU-Scope Metadata

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

TDVPS is described in the [TDX Module Spec].

5.3.1. UPDATED: 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.MNG.INITVP. 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 5.8: 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 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

5.3.2. How to Read the TDVPS (including TD VMCS) Tables

5.3.2.1. *UPDATED: Field Access*

The access columns describe whether this field is accessible to the host VMM in production mode (ATTRIBUTES.DEBUG == 0) and debug mode (ATTRIBUTES.DEBUG == 1) and to the guest TD. Access is done using the TDX module metadata access functions, e.g., TDH.VP.RD. Possible values are shown in the table below.

Table 5.9: Field Access Definition

Access	Meaning
None	No access to the field
RO	This field can only be read.
RW	This field can be read and written. No limitations are imposed except for checking that the value fits in the field size.

Access	Meaning
RWS	This field can be read and written. Writing is subject to the same limitations as if the field was modified by the guest TD (for guest state fields) and/or other limitation as described per field.

5.3.3. TDVPS (excluding TD VMCS)

Note: The table below lists only TDVPS fields that may be accessed by the host VMM in either production or debug mode.

Table 5.10: TDVPS Definition

Class	Field	VMM Prod. Access	VMM Debug Access	Guest Access	Type	Field Size (Bytes)	Num Fields	Num Elem.	Elem. Size (Bytes)	Description	Base FIELD_ID (Hex)
Management	VCPU_STATE	None	RO	None	VCPU_STATE	1	1	1	1	The activity state of the VCPU.	0xA020000000000000
Management	LAST_TD_EXIT	None	RO	None	LAST_TD_EXIT	1	1	1	1	Type of the last TD exit	0xA02000000000000F
Management	LAUNCHED	None	RO	None	Boolean	1	1	1	1	A Boolean flag, indicating whether the TD VCPU has been VMLAUNCH'ed on this LP since it has last been associated with this VCPU. If TRUE, VM entry should use VMRESUME. Else, VM entry should use VMLAUNCH.	0xA020000000000001
Management	VCPU_INDEX	RO	RO	RW	32b Unsigned Integer	4	1	1	4	Sequential index of the VCPU in the parent TD. VCPU_INDEX indicates the order of VCPU initialization (by TDINITVP), starting from 0, and is made available to the TD via TDINFO. VCPU_INDEX is in the range 0 to (TDCS.MAX_VCPUS - 1), up to 0xFFFE	0xA020000200000002
Management	NUM_TDCX	RO	RO	None	Unsigned Integer	1	1	1	1	Number of TDCX pages added to this TDVPS	0xA020000000000003
Management	TDVPS_PAGE_PA	RO	RO	None	Array of PA	8	6	1	8	An array of TDVPS_PAGES physical address pointers to the TDVPS physical pages • PA is without HKID bits • Page 0 is the PA of the TDVPR page • Pages 1,2... are PAs of the TDCX pages	0xA020000300000010
Management	ASSOC_LPID	RO	RO	None	Integer	4	1	1	4	The unique, hardware-derived identifier of the logical processor on which this VCPU is currently associated (either by TDENTER or by other VCPU-specific SEAMCALL flow): • A value of -1 indicates that VCPU is not associated with any LP. • Initialized by TDH.VP.INIT to the LP_ID on which it ran.	0xA020000200000004
Management	ASSOC_HKID	RO	RO	None	Integer	4	1	1	4	The TD's ephemeral private HKID at the last time this VCPU was associated (either by TDENTER or by other VCPU-specific SEAMCALL flow) with an LP: initialized by TDH.VP.INIT to the current TD ephemeral private HKID	0xA020000200000005
Management	VCPU_EPOCH	RO	RO	None	Integer	8	1	1	8	The value of TDCS.TD_EPOCH at the time this VCPU entered TDX non-root mode	0xA020000300000006
Management	CPUID_SUPERVISOR_VE	RO	RO	RW	Boolean	1	1	1	1	When set, the Intel TDX module injects #VE on guest TD execution of CPUID in CPL = 0.	0xA020000000000007
Management	CPUID_USER_VE	RO	RO	RW	Boolean	1	1	1	1	When set, the Intel TDX module injects #VE on guest TD execution of CPUID in CPL > 0.	0xA020000000000008
Management	IS_SHARED_EPTP_VALID	RO	RO	None	Boolean	1	1	1	1	Indicates that Shared EPTP is valid: set on successful TDH.VP.WR to Shared EPTP	0xA020000000000009
Management	LAST_EXIT_TSC	None	RO	None	Unsigned 64b Integer	8	1	1	8	Initialized to the value returned rdtsc on TDH.VP.INIT	0xA02000030000000A
Management	PEND_NMI	RW	RW	None	Boolean	1	1	1	1	When set, the Intel TDX module injects an NMI to the guest TD at the next available opportunity (NMI window open after TDENTER). the Intel TDX module then clears PEND_NIM.	0x202000000000000B
Management	XFAM	RO	RW	None	Bitmap	8	1	1	8	Copied from TDCS on TDH.VP.INIT. On TDH.VP.WR, checked for architectural and platform compatibility	0x202000030000000C
Management	LAST_EPF_GPA_LIST_IDX	None	RO	None	Unsigned Integer	1	1	1	1	Number of valid entries in LAST_EPF_GPA_LIST	0xA02000000000000D
Management	POSSIBLY_EPF_STEPPING	None	RO	None	Unsigned Integer	1	1	1	1	Number of possibly legal EPT Faults (EPFs) detected so far at this TD vCPU instruction	0xA02000000000000E
Management	LAST_EPF_GPA_LIST	None	RO	None	GPA	8	32	1	8	Array of GPAs that caused EPF so far at this TD vCPU instruction	0xA020000300000100
CPUID Control	CPUID_CONTROL	None	RO	RW	Array of 8-bit bitmaps	1	128	1	1	Bit 0: When set, the Intel TDX module injects #VE on guest TD execution of CPUID in CPL = 0.	0xA120000000000000

Class	Field	VMM Prod. Access	VMM Debug Access	Guest Access	Type	Field Size (Bytes)	Num Fields	Num Elem.	Elem. Size (Bytes)	Description	Base FIELD_ID (Hex)
										Bit 1: When set, the Intel TDX module injects #VE on guest TD execution of CPUID in CPL > 0. Other: Reserved, must be 0.	
TD VMCS	TD_VMCS	None	None	None	Page	1	4096	1	1		0x00200000FFFFFFFF
VAPIC	VAPIC	None	RO	None	Page	8	0	1	8	Virtual APIC Page	0x0120000300000000
VE_INFO	EXIT_REASON	None	RO	None	0	4	1	1	4		0x0220000200000000
VE_INFO	VALID	None	RO	None	0	4	1	1	4	0xFFFFFFFF: valid 0x00000000: not valid	0x0220000200000001
VE_INFO	EXIT_QUALIFICATION	None	RO	None	0	8	1	1	8		0x0220000300000002
VE_INFO	GLA	None	RO	None	0	8	1	1	8		0x0220000300000003
VE_INFO	GPA	None	RO	None	0	8	1	1	8		0x0220000300000004
VE_INFO	EPTP_INDEX	None	RO	None	0	2	1	1	2		0x0220000100000005
VE_INFO	INSTRUCTION_LENGTH	None	RO	None	0	4	1	1	4		0x8220000200000010
VE_INFO	INSTRUCTION_INFORMATION	None	RO	None	0	4	1	1	4		0x8220000200000011
Guest GPR State	RAX	None	RW	None	0	8	1	1	8		0x1020000300000000
Guest GPR State	RCX	None	RW	None	0	8	1	1	8	Init value is provided as an input to TDH.VP.INIT (same value as R8)	0x1020000300000001
Guest GPR State	RDX	None	RW	None	0	8	1	1	8	Init Value: • Bits [31:00]: Same as RESET value, matches CPUID.1:EAX. CPU version information includes Family, Model and Stepping • Bits [63:32]: Set to 0	0x1020000300000002
Guest GPR State	RBX	None	RW	None	0	8	1	1	8	Init Value: • Bits [05:00]: GPAW is 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 • Bits [63:06]: Reserved for future additional details, set to 0, must be ignored by vBIOS	0x1020000300000003
Guest GPR State	RSP_PLACEHOLDER	None	None	None	0	8	1	1	8		0x1020000300000004
Guest GPR State	RBP	None	RW	None	0	8	1	1	8	Init Value: • Bits [31:00]: Virtual CPU index, starting from 0 and allocated sequentially on each successful TDH.VP.INIT • Bits [63:32]: Set to 0	0x1020000300000005
Guest GPR State	RSI	None	RW	None	0	8	1	1	8		0x1020000300000006
Guest GPR State	RDI	None	RW	None	0	8	1	1	8	Init value is provided as an input to TDH.VP.INIT (same value as RCX)	0x1020000300000007
Guest GPR State	R8	None	RW	None	0	8	1	1	8		0x1020000300000008
Guest GPR State	R9	None	RW	None	0	8	1	1	8		0x1020000300000009
Guest GPR State	R10	None	RW	None	0	8	1	1	8		0x102000030000000A
Guest GPR State	R11	None	RW	None	0	8	1	1	8		0x102000030000000B
Guest GPR State	R12	None	RW	None	0	8	1	1	8		0x102000030000000C
Guest GPR State	R13	None	RW	None	0	8	1	1	8		0x102000030000000D
Guest GPR State	R14	None	RW	None	0	8	1	1	8		0x102000030000000E
Guest GPR State	R15	None	RW	None	0	8	1	1	8		0x102000030000000F
Guest State	DR0	None	RW	None	0	8	1	1	8		0x1120000300000000
Guest State	DR1	None	RW	None	0	8	1	1	8		0x1120000300000001
Guest State	DR2	None	RW	None	0	8	1	1	8		0x1120000300000002
Guest State	DR3	None	RW	None	0	8	1	1	8		0x1120000300000003
Guest State	DR6	None	RW	None	0	8	1	1	8		0x1120000300000006
Guest State	XCR0	None	RO	None	0	8	1	1	8		0x1120000300000020
Guest State	VCPU_STATE_DETAILS	ROS	ROS	None	0	8	1	1	8	Bit 0: VMXIP, indicates that a virtual interrupt is pending delivery, i.e. VMCS.RVI[7:4] > TDVPS.VAPIC.VPPR[7:4] Bits 63:1: Reserved, set to 0	0x9120000300000021
Guest MSR State	IA32_SPEC_CTRL	None	RW	None	0	8	1	1	8		0x1320000300000048
Guest MSR State	IA32_UMWAIT_CONTROL	None	RW	None	0	8	1	1	8		0x132000030000000E1

Class	Field	VMM Prod. Access	VMM Debug Access	Guest Access	Type	Field Size (Bytes)	Num Fields	Num Elem.	Elem. Size (Bytes)	Description	Base FIELD_ID (Hex)
Guest MSR State	IA32_PERFEVTSELx	None	RW	None	0	8	8	1	8		0x1320000300000186
Guest MSR State	MSR_OFFCORE_RSPx	None	RW	None	0	8	2	1	8		0x13200003000001A6
Guest MSR State	IA32_XFD	None	RO	None	0	8	1	1	8		0x13200003000001C4
Guest MSR State	IA32_XFD_ERR	None	RO	None	0	8	1	1	8		0x13200003000001C5
Guest MSR State	IA32_FIXED_CTRx	None	RW	None	0	8	4	1	8		0x1320000300000309
Guest MSR State	IA32_PERF_METRICS	None	RW	None	0	8	1	1	8		0x1320000300000329
Guest MSR State	IA32_FIXED_CTR_CTRL	None	RW	None	0	8	1	1	8		0x132000030000038D
Guest MSR State	IA32_PERF_GLOBAL_STATUS	None	RO	None	0	8	1	1	8		0x132000030000038E
Guest MSR State	IA32_PEBS_ENABLE	None	RW	None	0	8	1	1	8		0x13200003000003F1
Guest MSR State	MSR_PEBS_DATA_CFG	None	RW	None	0	8	1	1	8		0x13200003000003F2
Guest MSR State	MSR_PEBS_LD_LAT	None	RW	None	0	8	1	1	8		0x13200003000003F6
Guest MSR State	MSR_PEBS_FRONTEND	None	RW	None	0	8	1	1	8		0x13200003000003F7
Guest MSR State	IA32_A_PMCx	None	RW	None	0	8	8	1	8		0x13200003000004C1
Guest MSR State	IA32_DS_AREA	None	RW	None	0	8	1	1	8		0x1320000300000600
Guest MSR State	IA32_XSS	None	RO	None	0	8	1	1	8		0x1320000300000DA0
Guest MSR State	IA32_LBR_DEPTH	None	RW	None	0	8	1	1	8		0x13200003000014CF
Guest MSR State	IA32_STAR	None	RO	None	0	8	1	1	8		0x1320000300002081
Guest MSR State	IA32_LSTAR	None	RO	None	0	8	1	1	8		0x1320000300002082
Guest MSR State	IA32_FMASK	None	RO	None	0	8	1	1	8		0x1320000300002084
Guest MSR State	IA32_KERNEL_GS_BASE	None	RO	None	0	8	1	1	8		0x1320000300002102
Guest MSR State	IA32_TSC_AUX	None	RW	None	0	8	1	1	8		0x1320000300002103
Guest Ext. State	XBUFF	None	RW	None	XSAVES buffer	8	1536	1	8		0x1220000300000000

5.3.4. TD VMCS

Intel SDM, 24 Virtual Machine Control Structures

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

TD VMCS is a VMX format VMCS (with TDX ISA extensions) that is stored as part of TDVPS.

5.3.4.1. *TD VMCS Guest State Area*

5.3.4.1.1. TD VMCS Guest Register State Area

Intel SDM, Vol. 3, 9.1.1 Processor State after Reset
Intel SDM, Vol. 3, 24.4.1 Guest Register State

Table 5.11: TD VMCS Guest Register State Area Fields

Field	VMM Access		Init Value (after TDH.VP.INIT)
	Prod.	Debug	
Guest CR0	None	RWS	0x0021 • Bits PE (0) and NE (5) are set to 1. • All other bits are cleared to 0. The initial value is checked for compatibility with fixed-0 and fixed-1 bits according to IA32_VMX_CR0_FIXED* MSRs, except for PG (bit 31) which is allowed to be 0 since the guest TD runs as an unrestricted guest.
Guest CR3	None	RW	0

Field	VMM Access		Init Value (after TDH.VP.INIT)
	Prod.	Debug	
Guest CR4	None	RWS	0x2040 • Bits MCE (6) and VMXE (13) are set to 1 • All other bits are cleared to 0. The initial value is checked for compatibility with fixed-0 and fixed-1 bits according to IA32_VMX_CR4_FIXED* MSRs.
Guest DR7	None	RW	0x00000400
Guest RSP	None	RW	0
Guest RIP	None	RW	0xFFFFFFF0
Guest RFLAGS	None	RW	0x00000002
Guest ES selector	None	RW	0
Guest CS selector	None	RW	0
Guest SS selector	None	RW	0
Guest DS selector	None	RW	0
Guest FS selector	None	RW	0
Guest GS selector	None	RW	0
Guest LDTR selector	None	RW	0
Guest TR selector	None	RW	0
Guest ES base	None	RW	0
Guest CS base	None	RW	0
Guest SS base	None	RW	0
Guest DS base	None	RW	0
Guest FS base	None	RW	0
Guest GS base	None	RW	0
Guest LDTR base	None	RW	0
Guest TR base	None	RW	0
Guest GDTR base	None	RW	0
Guest IDTR base	None	RW	0
Guest ES limit	None	RW	0xFFFFFFFF
Guest CS limit	None	RW	0xFFFFFFFF
Guest SS limit	None	RW	0xFFFFFFFF
Guest DS limit	None	RW	0xFFFFFFFF
Guest FS limit	None	RW	0xFFFFFFFF
Guest GS limit	None	RW	0xFFFFFFFF
Guest LDTR limit	None	RW	0x0000FFFF
Guest TR limit	None	RW	0x0000FFFF
Guest GDTR limit	None	RW	0x0000FFFF
Guest IDTR limit	None	RW	0
Guest ES access rights	None	RW	0x0000C093 (Data, RW, Accessed, DPL=0, Present, 32b, 4KB granularity)
Guest CS access rights	None	RW	0x0000C09B (Code, RX, Accessed, DPL=0, Present, 32b)
Guest SS access rights	None	RW	0x0000C093 (Data, RW, Accessed, DPL=0, Present, 32b, 4KB granularity)
Guest DS access rights	None	RW	0x0000C093 (Data, RW, Accessed, DPL=0, Present, 32b, 4KB granularity)
Guest FS access rights	None	RW	0x0000C093 (Data, RW, Accessed, DPL=0, Present, 32b, 4KB granularity)
Guest GS access rights	None	RW	0x0000C093 (Data, RW, Accessed, DPL=0, Present, 32b, 4KB granularity)
Guest LDTR access rights	None	RW	0x00010082 (LDT, Present, 32b, 1B granularity, Unusable)
Guest TR access rights	None	RW	0x0000008B (32b TSS, Busy, Present, 32b, 1B granularity)
Guest SMBASE	None	None	0

5.3.4.1.2. TD VMCS Guest MSRs

See also the MSR virtualization tables in 2.1.

Table 5.12: TD VMCS Guest MSRs

Field	VMM Access		Init Value (after TDH.VP.INIT)
	Prod.	Debug	
IA32_DEBUGCTL	None	RWS	0
IA32_SYSENTER_CS	None	RW	0
IA32_SYSENTER_ESP	None	RW	0
IA32_SYSENTER_EIP	None	RW	0
IA32_PERF_GLOBAL_CTRL	None	RW	0x000000FF • EN_PMCx (bits 0 to (NUM_PMC - 1)) are set to 1. • All other bits are cleared to 0.
IA32_PAT	None	RW	0x0007040600070406
IA32_EFER	None	RW	0x901 • SCE (bit 0) is set to 1. • LME (bit 8) is set to 1. • NXE (bit 11) is set to 1. • All other bits are cleared to 0.
GUEST_IA32_S_CET	None	RW	0
GUEST_SSP	None	RW	0
GUEST_IA32_INTERRUPT_SSP_TABLE_ADDR	None	RW	0
IA32_RTIT_CTL	None	RW	0
IA32_LBR_CTL	None	RW	0
IA32_BNDCFGS	None	RO	0
IA32_GUEST_PKRS	None	RW	0

5.3.4.1.3. TD VMCS Guest Non-Register State Area

Intel SDM, 24.4.2 Guest Non-Register State

Table 5.13: TD VMCS Guest Non-Register State Area Fields

Field Name	VMM Access		Description	Initial State
	Prod.	Debug		
Activity State	None	RO	Saved/restored on VM exit/entry	Active (0)
Interruptibility State	None	RW	Saved/restored on VM exit/entry	0
Pending Debug Exceptions	None	RW	Saved/restored on VM exit/entry	0
VMCS Link Pointer	None	None	Saved/restored on VM exit/entry	NULL_PA (-1)
VMX-Preemption Timer Value	None	RW	N/A: VMX-preemption timer is not used by guest TDs.	0
PDPTEn	None	RW	N/A: PAE paging is not used by TD guests.	NULL_PA (-1)
Guest Interrupt Status	None	RW	Includes RVI (lower byte) and SVI (upper byte): saved/restored on VM exit/entry	0
PML Index	None	RW	N/A: PML is not used by guest TDs.	0
Guest UINV	None	RW		0

5.3.4.2. *TD VMCS Host State Area*

Intel SDM, 24.5 Host-State Area

The host state area is not intended to be accessible outside the Intel TDX module.

5.3.4.3. *TD VMCS VM-Execution Control Fields*

Intel SDM, 24.6 VM-Execution Control Fields

5.3.4.3.1. TD VMCS Pin-Based VM-Execution Controls

Table 5.14: TD VMCS Pin-Based VM-Execution Controls

		VMM Access			
Bit	**Name**	**Prod.**	**Debug**	**Description**	**Init Value**
0	External-interrupt exiting	None	RO	The Intel TDX module performs TD Exit	1
1	Reserved	None	RO		MSR
2	Reserved	None	RO		MSR
3	NMI exiting	None	RO	The Intel TDX module performs TD Exit	1
4	Reserved	None	RO		MSR
5	Virtual NMIs	None	RO		1
6	Activate VMX-preemption timer	None	RO		0
7	Process posted interrupts	RWS	RWS	Set to 1 by TDH.VP.WR only if a valid posted interrupt descriptor and a valid posted interrupt notification vector are set.	0
8	Reserved	None	RO		MSR
9	Reserved	None	RO		MSR
10	Reserved	None	RO		MSR
11	Reserved	None	RO		MSR
12	Reserved	None	RO		MSR
13	Reserved	None	RO		MSR
14	Reserved	None	RO		MSR
15	Reserved	None	RO		MSR
16	Reserved	None	RO		MSR
17	Reserved	None	RO		MSR
18	Reserved	None	RO		MSR
19	Reserved	None	RO		MSR
20	Reserved	None	RO		MSR
21	Reserved	None	RO		MSR
22	Reserved	None	RO		MSR
23	Reserved	None	RO		MSR
24	Reserved	None	RO		MSR
25	Reserved	None	RO		MSR
26	Reserved	None	RO		MSR
27	Reserved	None	RO		MSR
28	Reserved	None	RO		MSR
29	Reserved	None	RO		MSR
30	Reserved	None	RO		MSR
31	Reserved	None	RO		MSR

5

Reserved bits are set based on IA32_VMX_TRUE_PINBASED_CTLS MSR.

5.3.4.3.2. TD VMCS Processor-Based VM-Execution Controls

Table 5.15: TD VMCS Primary Processor-Based VM-Execution Controls

		VMM Access			
Bit	**Name**	**Prod.**	**Debug**	**Description**	**Init Value**
0	Reserved	None	RO		MSR
1	Reserved	None	RO		MSR
2	Interrupt-window exiting	None	RW		0
3	Use TSC offsetting	None	RO		1
4	Reserved	None	RO		MSR
5	Reserved	None	RO		MSR
6	Reserved	None	RO		MSR

Bit	Name	VMM Access		Description	Init Value
		Prod.	Debug		
7	HLT exiting	None	RO	The Intel TDX module injects a #VE into the guest TD	1
8	Reserved	None	RO		MSR
9	INVLPG exiting	None	RW		0
10	MWAIT exiting	None	RO	The Intel TDX module injects a #VE into the guest TD	1
11	RDPMC exiting	None	RW		~TDCS.ATTRIBUTES.PERFMON
12	RDTSC exiting	None	RW		0
13	Reserved	None	RO		MSR
14	Reserved	None	RO		MSR
15	CR3-load exiting	None	RW		0
16	CR3-store exiting	None	RW		0
17	Activate tertiary controls	None	RO		1
18	Reserved	None	RO		MSR
19	CR8-load exiting	None	RW		0
20	CR8-store exiting	None	RW		0
21	Use TPR shadow	None	RO		1
22	NMI-window exiting	None	RO	Set by the Intel TDX module before entering the guest TD – based on TDVPS.PEND_NMI	0
23	MOV-DR exiting	None	RW		0
24	Unconditional I/O exiting	None	RW		1
25	Use I/O bitmaps	None	RO		0
26	Reserved	None	RO		MSR
27	Monitor trap flag	None	RW		0
28	Use MSR bitmaps	None	RO		1
29	MONITOR exiting	None	RW		1
30	PAUSE exiting	None	RW		0
31	Activate secondary controls	None	RO		1

Reserved bits are set based on IA32_VMX_TRUE_PROCBASED_CTLS MSR.

Table 5.16: TD VMCS Secondary Processor-Based VM-Execution Controls

Bit	Name	VMM Access		Description	Init Value
		Prod.	Debug		
0	Virtualize APIC accesses	None	RO		0
1	Enable EPT	None	RO		1
2	Descriptor-table exiting	None	RW		0
3	Enable RDTSCP	None	RO		1
4	Virtualize x2APIC mode	None	RO		1
5	Enable VPID	None	RO		1
6	WBINVD exiting	None	RO		1
7	Unrestricted guest	None	RO		1
8	APIC-register virtualization	None	RO		1
9	Virtual-interrupt delivery	None	RO		1
10	PAUSE-loop exiting	None	RW		0
11	RDRAND exiting	None	RW		0
12	Enable INVPCID	None	RO		1
13	Enable VM functions	None	RO		1
14	VMCS shadowing	None	RO		0
15	Enable ENCLS exiting	None	RO		1
16	RDSEED exiting	None	RW		0
17	Enable PML	None	RWS	If set to 1, PML address must be a valid shared physical address	0
18	EPT-violation #VE	None	RO		1

		VMM Access			
Bit	**Name**	**Prod.**	**Debug**	**Description**	**Init Value**
19	Conceal VMX from PT	None	RO		1
20	Enable XSAVES/XRSTORS	None	RW		1
21	PASID translation	None	RO		1
22	Mode-based execute control for EPT	None	RO		0
23	Enable SPP	None	RO		0
24	PT uses guest physical addresses (PT2GPA)	None	RO		1
25	Use TSC scaling	None	RO		1
26	Enable user-level wait and pause	None	RO		Set to the value of virtualized CPUID(0x7,0x0).ECX[5]
27	Enable PCONFIG	None	RO		Set to the value of virtualized CPUID(0x7,0x0).EDX[18]
28	Enable ENCLV exiting	None	RO		1
29	Enable EPC Virtualization Extensions	None	RO		0
30	Bus-lock detection	RW	RW	If enabled by the host VMM (using TDH.VP.WR), then the Intel TDX module performs TD Exit on VM exit.	0
31	Notification exiting	RW	RW	If enabled by the host VMM (using TDH.VP.WR), then the Intel TDX module performs TD Exit on VM exit.	0

Table 5.17: TD VMCS Tertiary Processor-Based VM-Execution Controls

		VMM Access			
Bit	**Name**	**Prod.**	**Debug**	**Description**	**Init Value (after TDH.VP.INIT)**
0	LOADIWKEY exiting	None	RW		0
1	Enable HLAT	None	RO		0
2	EPT paging-write control	None	RO		0
3	Guest paging verification	None	RO		0
4	IPI virtualization	None	RO		0
5	GPAW	None	RO	0: GPA.SHARED bit is GPA[47] 1: GPA.SHARED bit is GPA[51]	Copied from TDCS.GPAW
6	Reserved	None	RO		MSR
7	Reserved	None	RO		MSR
8	Reserved	None	RO		MSR
9	Reserved	None	RO		MSR
10	Reserved	None	RO		MSR
11	Reserved	None	RO		MSR
12	Reserved	None	RO		MSR
13	Reserved	None	RO		MSR
14	Reserved	None	RO		MSR
15	Reserved	None	RO		MSR
16	Reserved	None	RO		MSR
17	Reserved	None	RO		MSR
18	Reserved	None	RO		MSR
19	Reserved	None	RO		MSR
20	Reserved	None	RO		MSR
21	Reserved	None	RO		MSR
22	Reserved	None	RO		MSR
23	Reserved	None	RO		MSR
24	Reserved	None	RO		MSR

Bit	Name	VMM Access		Description	Init Value (after TDH.VP.INIT)
		Prod.	Debug		
25	Reserved	None	RO		MSR
26	Reserved	None	RO		MSR
27	Reserved	None	RO		MSR
28	Reserved	None	RO		MSR
29	Reserved	None	RO		MSR
30	Reserved	None	RO		MSR
31	Reserved	None	RO		MSR
32	Reserved	None	RO		MSR
33	Reserved	None	RO		MSR
34	Reserved	None	RO		MSR
35	Reserved	None	RO		MSR
36	Reserved	None	RO		MSR
37	Reserved	None	RO		MSR
38	Reserved	None	RO		MSR
39	Reserved	None	RO		MSR
40	Reserved	None	RO		MSR
41	Reserved	None	RO		MSR
42	Reserved	None	RO		MSR
43	Reserved	None	RO		MSR
44	Reserved	None	RO		MSR
45	Reserved	None	RO		MSR
46	Reserved	None	RO		MSR
47	Reserved	None	RO		MSR
48	Reserved	None	RO		MSR
49	Reserved	None	RO		MSR
50	Reserved	None	RO		MSR
51	Reserved	None	RO		MSR
52	Reserved	None	RO		MSR
53	Reserved	None	RO		MSR
54	Reserved	None	RO		MSR
55	Reserved	None	RO		MSR
56	Reserved	None	RO		MSR
57	Reserved	None	RO		MSR
58	Reserved	None	RO		MSR
59	Reserved	None	RO		MSR
60	Reserved	None	RO		MSR
61	Reserved	None	RO		MSR
62	Reserved	None	RO		MSR
63	Reserved	None	RO		MSR

Reserved bits are set based on IA32_VMX_PROCBASED_CTLS3 MSR.

5.3.4.3.3. TD VMCS Controls for APIC Virtualization

Table 5.18: TD VMCS Controls for APIC Virtualization

Field Name	VMM Access		Description	Initial Value
	Prod.	Debug		
APIC-access address	None	RO		NULL_PA (-1)
Virtual-APIC address	None	None	On VCPU-to-LP association, set by the Intel TDX module to the address of the VAPIC page in TDVPS, including the TD's ephemeral HKID	Address of the VAPIC page in TDVPS, including the TD's ephemeral HKID

Field Name	VMM Access		Description	Initial Value
	Prod.	Debug		
TPR threshold	None	RO		0
EOI-exit bitmap n	None	RO		0
Posted-interrupt notification vector	RWS	RWS	TDH.VP.WR checks the value to be in the range 0 to 255. See process posted interrupt pin-based execution control.	0xFFFF
Posted-interrupt descriptor address	RWS	RWS	TDH.VP.WR checks the value as follows: • It must be a valid shared physical address (HKID bits encode a shared HKID). • It must be aligned on 64B. See process posted interrupt pin-based execution control.	0xFFFFFFFFFFFFFFC0

5.3.4.3.4. EPTP and Shared EPTP

Table 5.19: EPTP (Copied from TDCS.EPTP on TDH.VP.INIT)

Bits	Field Name	VMM Access		Description	Initial Value
		Prod.	Debug		
2:0	EPT Memory Type	RO	RO	Set to WB	6
5:3	EPT Level	RO	RO	1 less than the EPT page-walk length	Copied from TDCS.EPTP
6	Enable A/D Bits	RO	RO		0
7	Enable supervisor shadow stack control	RO	RO		0
11:8	Reserved	RO	RO		0
51:12	EPML5/4 PA	RO	RO		
63:52	Reserved	RO	RO		0

Table 5.20: Shared EPTP

Bits	Field Name	VMM Access		Description	Initial Value
		Prod.	Debug		
11:0	Reserved	None	RO		0
51:12	EPML5/4 PA	RWS	RWS		
63:52	Reserved	None	RO		0

5.3.4.3.5. CR-Related TD VMCS VM-Execution Control Fields

Table 5.21: CR-Related VMCS VM-Execution Control Fields

Field Name	VMM Access		Description	Initial Value
	Prod.	Debug		
CR0 Guest/Host Mask	None	RW	Bits 0, 5, 29 and 30 can't be written even in debug mode	The following bits are set to 1, indicating they are owned by the Intel TDX module: • PE (0) • NE (5) • NW (29) • CD (30) • Any bit set to 1 in IA32_VMX_CR0_FIXED0 (i.e., a bit whose value must be 1) • Any bit set to 0 in IA32_VMX_CR0_FIXED1 (i.e., a bit whose value must be 0), except for PG(31) which is set to 0, since the guest TD runs as an unrestricted guest • Bits known to the Intel TDX module as reserved (bits 63-32, 28-19, 17 and 15-6) All other bits are cleared to 0, indicating they are owned by the guest TD.
CR0 Read Shadow	None	RW	Bits 0 and 5 can't be written even in debug mode	The following bits are set to 1: • PE (0) • NE (5) • Any bit set to 1 in IA32_VMX_CR0_FIXED0 (i.e., a bit whose value must be 1) All other bits are cleared to 0.
CR4 Guest/Host Mask	None	RW	Bits 6, 13 and 14 can't be written even in debug mode	• Bits MCE (6), VMXE (13) and SMXE (14) are set to 1, indicating they are owned by the Intel TDX module. • If PK is not enabled, then bit PKE (22) is set to ~TDCS.XFAM[9] to intercept writes to CR4. • If TDCS.XFAM[12:11] is 11, then bit CET (23) is cleared to 0. Otherwise, if CET is not enabled, then bit CET (23) is set to 1 to intercept writes to CR4. • If ULI is not enabled, then bit UINT (25) is set ~TDCS.XFAM[14] to intercept writes to CR4. • If KeyLocker is not enabled, then bit KL (19) is set to ~TDCS.ATTRIBUTES.KL to intercept writes to CR4. • If PKS is not enabled, then bit PKS (24) is set to ~TDCS.ATTRIBUTES.PKS to intercept writes to CR4. • If Perfmon is not enabled, then bit PCE (8) is set to ~TDCS.ATTRIBUTES.PERFMON to intercept writes to CR4. • Any bit set to 1 in IA32_VMX_CR4_FIXED0 (i.e., a bit whose value must be 1) is set to 1. • Any bit set to 0 in IA32_VMX_CR4_FIXED1 (i.e., a bit whose value must be 0) is set to 1. • Bits known to the Intel TDX module as reserved (bits 63-26 and bit 15) are set to 1. • All other bits are cleared to 0.

Field Name	VMM Access		Description	Initial Value
	Prod.	Debug		
CR4 Read Shadow	None	RW	Bit 6 can't be written even in debug mode	• Bit MCE (6) is set to 1. • Bit VMXE (13) is cleared to 0. • Any other bit whose value is set to 1 in IA32_VMX_CR4_FIXED0 (i.e., a bit whose value must be 1) is set to 1. • All other bits are cleared to 0.
CR3-Target Values	None	RW	N/A: The Intel TDX module does not control guest CR3	N/A
CR3-Target Count	None	RW	Set to 0: Intel TDX module does not control guest CR3	0

5.3.4.3.6. Other TD VMCS VM-Execution Control Fields

Table 5.22: Other TD VMCS VM-Execution Control Fields

Field Name	VMM Access		Description	Initial Value
	Prod.	Debug		
Exception Bitmap	None	RW	• Bit 18 (MCE) is set to 1, even in debug mode. • Other bits are cleared to 0. They may be modified in debug mode.	0x00040000
Page-fault error-code mask	None	RW		0
Page-fault error-code match	None	RW		0
I/O-Bitmap Address n	None	RO	Set to NULL_PA (-1): I/O bitmaps execution control is set to 0	NULL_PA (-1)
Time-Stamp Counter Offset	RO	RW		Copied from TDCS.TSC_OFFSET
Time-Stamp Counter Multiplier	RO	RW		Copied from TDCS.TSC_MULTIPLIER
MSR-Bitmap Address	RO	RO		
Executive-VMCS Pointer	None	None	N/A	NULL_PA (-1)
TD HKID	RO	RO		
VPID	None	RO	1 + the sequential initialization index of the VCPU (TDVPS.VCPU_INDEX + 1)	Set to 1 + the sequential index of the VCPU (TDVPS.VCPU_INDEX + 1)
PLE_GAP	RO	RW		0
PLE_Window	RO	RW		0
VM-Function Controls	RO	RO	The Intel TDX module injects a #UD into the TD.	0
EPTP-list address	RO	RO	VMFUNC is not supported.	NULL_PA (-1)

Field Name	VMM Access		Description	Initial Value
	Prod.	Debug		
VMREAD-bitmap address	None	RO	VMCS shadowing is not supported.	NULL_PA (-1)
VMWRITE-bitmap address	None	RO	VMCS shadowing is not supported.	NULL_PA (-1)
ENCLS-Exiting Bitmap	None	RO	Set to all 1's – the Intel TDX module injects a #UD into the guest TD.	All 1s
ENCLV-Exiting Bitmap	None	RO	Set to all 1's – the Intel TDX module injects a #UD into the guest TD.	All 1s
PML address	RO	RWS	TDH.VP.WR checks the value as follows: • It must be a valid shared physical address (HKID bits encode a shared HKID). • It must be aligned on 4KB. See enable PML execution control.	0xFFFFFFFFFFFFF000
Virtualization-exception information address	None	RO		
EPTP index	None	RO		0
XSS-Exiting Bitmap	None	RW		0
low PASID directory address	None	RO		
high PASID directory address	None	RO		
notify window	RW	RW		0
PCONFIG-Exiting Bitmap	None	RO		-1

5.3.4.4. *TD VMCS VM-Exit Control Fields*

Intel SDM, 24.7 VM-Exit Control Fields

Table 5.23: TD VMCS VM-Exit Controls

	VMM Access			
Name	**Prod.**	**Debug**	**Description**	**Init Value**
Reserved	None	RO		MSR
Reserved	None	RO		MSR
Save debug controls	None	RO		1
Reserved	None	RO		MSR
Reserved	None	RO		MSR
Reserved	None	RO		MSR
Reserved	None	RO		MSR
Reserved	None	RO		MSR
Reserved	None	RO		MSR
Host address-space size	None	RO		1
Reserved	None	RO		MSR
Reserved	None	RO		MSR

	VMM Access			
Name	**Prod.**	**Debug**	**Description**	**Init Value**
Load IA32_PERF_GLOBAL_CTRL	None	RO		Set to 1 if TDCS.ATTRIBUTES.PERFMON = 1 or ATTRIBUTES.DEBUG = 1
Reserved	None	RO		MSR
Reserved	None	RO		MSR
Acknowledge interrupt on exit	None	RO		1
Reserved	None	RO		MSR
Reserved	None	RO		MSR
Save IA32_PAT	None	RO		1
Load IA32_PAT	None	RO		1
Save IA32_EFER	None	RO		1
Load IA32_EFER	None	RO		1
Save VMX-preemption time value	None	RO		Set to 1 if TDCS.ATTRIBUTES.DEBUG = 1
Clear IA32_BNDCFGS	None	RO	Deprecated	0
Conceal VMX from PT	None	RO		1
Clear IA32_RTIT_CTL	None	RO		1
Clear IA32_LBR_CTL	None	RO		1
Clear UINV	None	RO		1
Load host CET state	None	RO		1
Load host PKRS	None	RO		0
Save IA32_PERF_GLOBAL_CTRL	None	RO		Set to 1 if TDCS.ATTRIBUTES.PERFMON = 1 or ATTRIBUTES.DEBUG = 1
Reserved	None	RO		MSR

Reserved bits are set based on IA32_VMX_TRUE_EXIT_CTLS MSR.

Table 5.24: TD VMCS VM-Exit Controls for MSRs

Field Name	**VMM Access**		**Description**	**Initial Value**
	Prod.	**Debug**		
VM-exit MSR-store count	None	RO	Not used	0
VM-exit MSR-store address	None	RO	Not used	NULL_PA (-1)
VM-exit MSR-load count	None	RW	Not used	0
VM-exit MSR-load address	None	RO	Not used	NULL_PA (-1)

5 #### 5.3.4.5. *TD VMCS VM-Entry Control Fields*

Intel SDM, 24.8 VM-Entry Control Fields

Table 5.25: TD VMCS VM-Entry Controls

		VMM Access			
Bit	**Name**	**Prod.**	**Debug**	**Description**	**Init Value (after TDH.VP.INIT)**
0	Reserved	None	RO		MSR
1	Reserved	None	RO		MSR
2	Load debug controls	None	RO		1
3	Reserved	None	RO		MSR
4	Reserved	None	RO		MSR
5	Reserved	None	RO		MSR
6	Reserved	None	RO		MSR
7	Reserved	None	RO		MSR
8	Reserved	None	RO		MSR
9	IA-32e mode guest	None	RO	Written by the CPU on VM exit	0
10	Entry to SMM	None	RO		0

		VMM Access			
Bit	**Name**	**Prod.**	**Debug**	**Description**	**Init Value (after TDH.VP.INIT)**
11	Deactivate dual-monitor treatment	None	RO		0
12	Reserved	None	RO		MSR
13	Load IA32_PERF_GLOBAL_CTRL	None	RO		Set to 1 if TDCS.ATTRIBUTES.PERFMON = 1 or ATTRIBUTES.DEBUG = 1
14	Load IA32_PAT	None	RO		1
15	Load IA32_EFER	None	RO		1
16	Load IA32_BNDCFGS	None	RO		0
17	Conceal VMX from PT	None	RO		1
18	Load IA32_RTIT_CTL	None	RO		1
19	Load UINV	None	RO		1
20	Load CET state	None	RO		1
21	Load IA32_LBR_CTL	None	RO		1
22	Load guest PKRS	None	RO		Set to 1 if TDCS.ATTRIBUTES.PKRS = 1 or TDCS.ATTRIBUTES.DEBUG = 1
23	Reserved	None	RO		MSR
24	Reserved	None	RO		MSR
25	Reserved	None	RO		MSR
26	Reserved	None	RO		MSR
27	Reserved	None	RO		MSR
28	Reserved	None	RO		MSR
29	Reserved	None	RO		MSR
30	Reserved	None	RO		MSR
31	Reserved	None	RO		MSR

Reserved bits are set based on IA32_VMX_ENTRY_CTLS MSR.

Table 5.26: TD VMCS VM-Entry Controls for MSRs

Field Name	**VMM Access**		**Description**	**Initial Value**
	Prod.	**Debug**		
VM-entry MSR-load count	None	RO	Not used	0
VM-entry MSR-load address	None	RO	Not used	NULL_PA (-1)

Table 5.27: TD VMCS VM-Entry Controls for Event Injection

Field Name	**VMM Access**		**Description**	**Initial Value**
	Prod.	**Debug**		
VM-entry interruption information	None	RO		
VM-entry exception error code	None	RO		
VM-entry instruction length	None	RO		

5.3.4.6. *TD VMCS VM-Exit Information Fields*

Intel SDM, 24.9 VM-Exit Information Fields

Table 5.28: TD VMCS Basic VM-Exit Information

Field Name	VMM Access		Description	Initial Value
	Prod.	Debug		
Exit reason	None	RO	If the Intel TDX module decides to perform a TD exit, it returns this in RAX bits 31:0. Bit 27 (enclave mode) is not set. Bit 28 (Pending MTF VM exit) is not set. Bit 29 (VM exit from VMX root operation) is not set. Bit 31 (VM-entry failure) is not set.	N/A
Exit qualification	None	RO	If the Intel TDX module decides to perform a TD exit, it returns this in RCX. If the exit is due to EPT violation, bits 12-7 of the exit qualification are cleared to 0.	N/A
Guest-Linear Address	None	RO		N/A
Guest-physical Address	None	RO	If the Intel TDX module decides to perform a TD exit, it returns this in R8. It the EPT fault was caused by an access attempt to a private page, the Intel TDX module clears bits 11:0 to 0.	N/A

Table 5.29: TD VMCS Information for VM Exits Due to Vectored Events

Field Name	VMM Access		Description	Initial Value
	Prod.	Debug		
VM-exit interruption information	None	RO	On asynchronous TD exit, the Intel TDX module returns this in R9. Bits 63:32 are cleared to 0.	N/A
VM-exit interruption error code	None	RO		N/A

Table 5.30: TD VMCS Information for VM Exits That Occur During Event Delivery

Field Name	VMM Access		Description	Initial Value
	Prod.	Debug		
IDT-vectoring information	None	RO		
IDT-vectoring error code	None	RO		

Table 5.31: TD VMCS Information for VM Exits Due to Instruction Execution

Field Name	VMM Access		Description	Initial Value
	Prod.	Debug		
VM-exit instruction length	None	RO		
VM-exit instruction information	None	RO		

Field Name	VMM Access		Description	Initial Value
	Prod.	Debug		
I/O RCX	None	RO		N/A
I/O RSI	None	RO		N/A
I/O RDI	None	RO		N/A
I/O RIP	None	RO		N/A

Table 5.32: TD VMCS VM-Instruction Error Field

Field Name	VMM Access		Description	Initial Value
	Prod.	Debug		
VM-instruction error	None	RO		N/A

6. UPDATED: ABI Reference: Interface Functions

6.1. 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 6.1 below describes how to read it. Most of the background is detailed in the [TDX Module Spec].

Table 6.1: How to Read the Operands Information Tables

Explicit/ Implicit	Reg.	Addr. 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	HPA or GPA	Resource (memory or CPU internal) for this operand	Data type of the resource, as defined in Chapter 4 or Chapter 5	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 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. Shared(i) and Exclusive(i) indicate that the resource is implicitly restricted.		

6.2. NEW: Common Algorithms Used by Multiple Interface Functions

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

6.2.1. Metadata Access

6.2.1.1. 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.

6.2.1.2. *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 and TDG.VP.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 or a guest-side 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 an effective write mask:
   3.1. If a write mask is provided as an input, derive the effective write mask by bitwise-anding the write mask derived above with the write mask provided as an input.
   3.2. Else, the effective write mask is the write mask derived above.
4. If the effective 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. 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.
   6.1. 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:

7. Mask the old field value with the read mask derived above, and return the resulting value.

6.2.1.3. *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

Intel TDX Application Binary Interface (ABI) Reference

6.3. UPDATED: 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.

6.3.1. UPDATED: SEAMCALL Instruction (Common)

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

Table 6.2: SEAMCALL Input Operands Definition

Parameter	Description		
RAX	Leaf and version numbers, as defined in the [TDX Module Spec]. See Table 6.4 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 6.3: SEAMCALL Output Operands Definition

Parameter	Description
RAX	Instruction return code, indicating the outcome of execution of the instruction. See the [TDX Module Spec] for details.
Other	See individual SEAMCALL leaf functions.

Table 6.4: 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
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

Leaf #	Interface Function Name	Description
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	Shutdown the Intel TDX module on the current LP
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
64	TDH.EXPORT.ABORT	Abort an export session
65	TDH.EXPORT.BLOCKW	Block a TD private page for writing
66	TDH.EXPORT.RESTORE	Cancel the export of a previously exported TD private page
68	TDH.EXPORT.MEM	Export a TD private page
70	TDH.EXPORT.PAUSE	Pause the exported TD
71	TDH.EXPORT.TRACK	End the in-order export phase and generate a start token
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
82	TDH.IMPORT.COMMIT	Commit the import session and allow the imported TD to run
83	TDH.IMPORT.MEM	Import a TD private page
84	TDH.IMPORT.TRACK	Process a start token and end the in-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.

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 6.5: 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.

6.3.2. NEW: 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 6.6: TDH.EXPORT.ABORT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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	If an abort token is available, this is the migration stream index. Otherwise, this field's value must be 0.
	63:16	RESERVED	Reserved: must be 0

Table 6.7: TDH.EXPORT.ABORT Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
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.

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 6.8: TDH.EXPORT.ABORT Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	N/A	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(i)	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 lower than TDCS.NUM_MIGS.

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 1 as the direction bit, 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 6.9: TDH.EXPORT.ABORT Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
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_TD_NOT_INITIALIZED	

6.3.3. NEW: TDH.EXPORT.BLOCKW Leaf

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

Table 6.10: TDH.EXPORT.BLOCKW Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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_LIST_INFO: HPA of a GPA list page in shared memory, and first and last entries to process, as defined in 4.10.2		
RDX	HPA of the source TD's TDR page (HKID bits must be 0)		

Table 6.11: TDH.EXPORT.BLOCKW Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
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
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 marks it as blocked for writing (by 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.

List Entry Error: If a page can't be blocked for writing, TDH.EXPORT.BLOCKW 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.BLOCKW is interruptible. If a pending interrupt is detected during operation, TDH.EXPORT.BLOCKW returns with a TDX_INTERRUPED_RSUMABLE 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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.12: TDH.MEM.RANGE.BLOCKW Memory Operands Information

Explicit/ Implicit	Reg.	Addr. 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	R	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	Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	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).

f 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 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. Block the Secure EPT entry for writing. Atomically set its state to BLOCKEDW, PENDING_BLOCKEDW, EXPORTED_DIRTY_BLOCKEDW or PENDING_EXPORTED_DIRTY_BLOCKEDW as appropriate.
      6.4.3. 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 6.13: TDH.EXPORT.BLOCKW Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
TDX_EPT_ENTRY_FREE	
TDX_EPT_WALK_FAILED	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.EXPORT.BLOCKW is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_INITIALIZED	

6.3.4. NEW: 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 6.14: TDH.EXPORT.MEM Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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_LIST_INFO: HPA of a GPA list page in shared memory, and first and last entries to process, as defined in 4.10.2 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	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
R9	HPA (including HKID bits) of a migration buffer list in shared memory, corresponding to the GPA list pointed by RCX – see 4.10.3.		
R10	Migration stream and resume flag:		
	Bits	**Name**	**Description**
	15:0	MIGS_INDEX	Migration stream index
	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	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 4.10.3.		
R12	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 4.10.3.		

Table 6.15: TDH.EXPORT.MEM Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
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

Operand	Description
RDX	Number of exported 4KB migration buffers
AVX, AVX2 and AVX512 state	May be reset to the architectural INIT 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.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 and 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 to export the page or to cancel a previous page export. It is also possible to skip entries in the least by requesting no operation for specific entries. The GPA list format is described in 4.10.2. It is designed to be compatible with TDH.EXPORT.BLOCKW.

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

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_INTERRUPED_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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.16: TDH.EXPORT.MEM Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	RW	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	R	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	GPA	TD private pages (via GPA list)	Block	None	Private	4KB	None	None	None

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	N/A	HPA	Migration buffer pages (via page list)	Blob	RW	Shared	4KB	None	None	None
Explicit	N/A	HPA	Migration buffer pages (via page list)	Blob	RW	Shared	4KB	None	None	None
Explicit	N/A	HPA	MAC 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
Implicit	N/A	N/A	Migration context	N/A	RW	Opaque	N/A	None	N/A	N/A
Implicit	N/A	GPA	Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	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. If the export is in the in-order phase, the migration stream index is lower than TDCS.NUM_IN_ORDER_MIGS. Else, 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 0 as the direction bit, 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 SEPT 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.

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

10.8. Update the SEPT entry state, GPA list entry and migration buffer list entry.
10.9. Increment the migration stream context's IV_COUNTER
10.10. Build the 96b IV for this page by concatenating 0 as the direction bit, the stream index and the stream context's IV_COUNTER.
10.11. Accumulate page MAC based on the GPA list entry.
10.12. If the page content is to be exported, encrypt the TD private page into the migration buffer and accumulate MAC.
10.13. Write the page MAC to the MAC list.
10.14. If this is not the last round and there is a pending interrupt:
10.14.1. Save intermediate state in the migration stream context.
10.14.2. Terminate TDH.EXPORT.MEM with a TDX_INTERRUPTED_RESUMABLE status.
10.15. 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 6.17: TDH.EXPORT.MEM Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_OPERAND_PAGE_INVALID	
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_INITIALIZED	

6.3.5. NEW: TDH.EXPORT.PAUSE Leaf

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

Table 6.18: TDH.EXPORT.PAUSE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	HPA of Source TD TDR page (HKID bits must be 0)		

Table 6.19: TDH.EXPORT.PAUSE Output Operands Definitions

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
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.

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 6.20: TDH.EXPORT.PAUSE Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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 6.21: TDH.EXPORT.PAUSE Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
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_TD_NOT_INITIALIZED	

6.3.6. NEW: TDH.EXPORT.RESTORE Leaf

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

Table 6.22: TDH.EXPORT.RESTORE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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_LIST_INFO: HPA of a GPA list page in shared memory, and first and last entries to process, as defined in 4.10.2		
RDX	HPA of the source TD's TDR page (HKID bits must be 0)		

Table 6.23: TDH.EXPORT.RESTORE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
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
AVX, AVX2 and AVX512 state	May be reset to the architectural INIT 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.RESTORE restores a list of TD private 4KB pages' Secure EPT entry states after an aborted export session. It reverts each Secure EPT entry to its original non-exported state.

List Entry Error: If a page's Secure EPT entry can't be restored, TDH.EXPORT.RESTORE 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.RESTORE is interruptible. If a pending interrupt is detected during operation, TDH.EXPORT.RESTORE returns with a TDX_INTERRUPED_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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.24: TDH.EXPORT.RESTORE Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	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 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_CONT.
      6.4.3. If the SEPT state is one of the PENDING_* states, update it to PENDING. Else, update it to MAPPED.
   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 6.25: TDH.EXPORT.RESTORE Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_OPERAND_PAGE_INVALID	
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_INITIALIZED	

6.3.7. NEW: 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.

Table 6.26: TDH.EXPORT.STATE.IMMUTABLE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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
R9	PAGE_LIST_INFO: migration buffers list information – see 4.10.2.1		
R10	Migration stream and resume flag:		
	Bits	**Name**	**Description**
	15:0	MIGS_INDEX	Migration stream index – must be 0
	31:16	NUM_IN_ORDER_MIGS	Number of migration streams to be used during the in-order migration phase
	62:32	RESERVED	Reserved: must be 0
	63	RESUME	0: This is a new invocation 1: This is resumption of a previously interrupted operation

Table 6.27: TDH.EXPORT.STATE.IMMUTABLE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
RDX	Number of exported 4KB migration buffers
AVX, AVX2 and AVX512 state	May be reset to the architectural INIT 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 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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.28: TDH.EXPORT.STATE.IMMUTABLE Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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(i)	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.
7. Any previous aborted export session has been cleaned up: TDCS.MIG_COUNT is 0.
8. MIGS_INDEX is 0.
9. NUM_IN_ORDER_MIGS is in the range 1 through TDCS.NUM_MIGS.
10. The buffer provided for MBMD is large enough.
11. 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:

12. 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):
    12.1. Check that a valid migration 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.

    If passed:

    12.2. Initialize the migration context in TDCS:
        12.2.1. Copy the migration key to a working migration key that will be used throughout the export session.
        12.2.2. Copy NUM_IN_ORDER_MIGS to TDCS.
        12.2.3. Set all migration streams' INITIALIZED flags to 0 and ENABLED flags to 1.
    12.3. Initialize the current migration stream.
    12.4. Increment the migration stream context's IV_COUNTER.
    12.5. 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.
    12.6. Build the MBMD in the migration stream context.
    12.7. Accumulate MAC in the stream context based on the MAC'ed fields of MBMD.
13. Else (this is a resumption of a previously interrupted TDH.EXPORT.STATE.IMMUTABLE):
    13.1. Check that the resumption is valid:
        13.1.1. The stream context indicates there's a valid interruption state.
        13.1.2. The current SEAMCALL leaf number and the PAGE_OR_LIST operand have the same value as in the interruption state.
    13.2. Check that the migration stream is enabled.
    13.3. Restore the previously saved page list index from the migration context.
    13.4. 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.

If passed:

14. Repeat exporting 4KB pages until all immutable state is exported or until a pending interrupt is detected:
    14.1. Get the 4KB next page HPA from it from the page list.
    14.2. Dump the next set of metadata fields as a metadata list of field sequences, into an internal temporary 4KB buffer.
    14.3. 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.
    14.4. If all immutable state has been exported:
        14.4.1. Write the accumulated MAC to the MBMD in the stream context.
        14.4.2. Write the MBMD to the memory buffer provided by the host VMM.
        14.4.3. Mark the migration stream context's interrupted state as invalid.
        14.4.4. Increment the migration stream context's NEXT_MB_COUNTER.
        14.4.5. Set TDCS.TOTAL_MB to 1.
        14.4.6. Set TDCS.OP_STATE to LIVE_EXPORT.
        14.4.7. Clear TDCS.DIRTY_COUNT to 0.
        14.4.8. Terminate TDH.EXPORT.STATE.IMMUTABLE with a TDX_SUCCESS status.
    14.5. Else, if there is a pending interrupt:
        14.5.1. Save the interruption state to the stream context
        14.5.2. Terminate TDH.EXPORT.STATE.IMMUTABLE with a TDX_INTERRUPTED_RESUMABLE status.

Completion Status Codes

Table 6.29: TDH.EXPORT.STATE.IMMUTABLE Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	
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_TD_NOT_INITIALIZED	

6.3.8. NEW: TDH.EXPORT.STATE.TD Leaf

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

Table 6.30: TDH.EXPORT.STATE.TD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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
R9	PAGE_LIST_INFO: migration buffers list information – see 0		
R10	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 6.31: TDH.EXPORT.STATE.TD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
RDX	Number of exported 4KB migration buffers
AVX, AVX2 and AVX512 state	May be reset to the architectural INIT 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.

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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.32: TDH.EXPORT.STATE.TD Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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
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	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	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.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 0 as the direction bit, 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 6.33: TDH.EXPORT.STATE.TD Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	
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_TD_NOT_INITIALIZED	

6.3.9. NEW: TDH.EXPORT.STATE.VP Leaf

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

Table 6.34: TDH.EXPORT.STATE.VP Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	HPA of the source TD VCPU's TDVPR 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
R9	PAGE_LIST_INFO: migration buffers list information – see 0		
R10	Migration stream and resume flag:		
	Bits	**Name**	**Description**
	15:0	MIGS_INDEX	Migration stream index
	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 6.35: TDH.EXPORT.STATE.VP Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
RDX	Number of exported 4KB migration buffers
AVX, AVX2 and AVX512 state	May be reset to the architectural INIT 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.

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 Spec].

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 6.36: TDH.EXPORT.STATE.VP Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDVPR page	TDVPS	R	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	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	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_IN_ORDER_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. 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):
    10.1. If the migration stream has not been initialized, initialize it.
    10.2. Increment the migration stream context's IV_COUNTER.
    10.3. Build the MBMD in the migration stream context.
    10.4. 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.
    10.5. 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.VP):
    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 TDVPR HAP and PAGE_OR_LIST operands are the same as in the interruption state.
    11.2. Increment the migration stream context's IV_COUNTER.
    11.3. Restore the previously saved page list index from the migration context.
12. Repeat exporting 4KB pages until all immutable 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 VCPU 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.VP 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.VP with a TDX_INTERRUPTED_RESUMABLE status.

Completion Status Codes

Table 6.37: TDH.EXPORT.STATE.VP Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	
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_TD_NOT_INITIALIZED	

6.3.10. NEW: 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 6.38: TDH.EXPORT.TRACK Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 6.39: TDH.EXPORT.TRACK Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
AVX, AVX2 and AVX512 state	May be reset to the architectural INIT 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.

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.

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 6.40: TDH.EXPORT.TRACK Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	N/A	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	None	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 R10.IN_ORDER_DONE is 0:
   9.1. Increment TDCS.MIG_EPOCH
10. Else (R10.IN_ORDER_DONE is 1):
    10.1. Check that an export session is in the in-order phase and the TD has been paused: TDCS.OP_STATE is PAUSED_EXPORT.
    10.2. 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).
    10.3. The TD mutable state has been exported (by TDH.EXPORT.STATE.TD).

    If passed:

    10.4. Start the out-of-order phase:
        10.4.1. Set TDCS.OP_STATE to POST_EXPORT.
        10.4.2. Set TDCS.MIG_EPOCH to 0xFFFFFFFF.
11. Increment the migration stream context's IV_COUNTER.
12. 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.
13. Create an epoch token MBMD with the following fields:
    13.1. The number of the new epoch that have just begun. Bit 63 indicates the beginning of the out-of-order phase.
    13.2. The total number of migration bundles (including the current one) that have been exported in the current migration session.

14. Accumulate MAC based on the MAC'ed fields of MBMD and write to the MBMD's MAC field's value.
15. Write the MBMD to the provided memory buffer.

Completion Status Codes

Table 6.41: TDH.EXPORT.TRACK Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
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_TD_NOT_INITIALIZED	

6.3.11. NEW: TDH.EXPORT.UNBLOCKW Leaf

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

Table 6.42: TDH.EXPORT.UNBLOCKW Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 4.5.1: 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 6.43: TDH.EXPORT.UNBLOCKW Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
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).

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 6.44: TDH.EXPORT.UNBLOCKW Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	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	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	Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	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 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.
11. Else (Secure EPT entry state is EXPORTED_DIRTY_BLOCKEDW or PENDING_EXPORTED_DIRTY_BLOCKEDW):
    11.1. Unblock the Secure EPT entry for writing by atomically setting its state to EXPORTED_DIRTY or PENDING_EXPORTED_DIRTY, respectively.
    11.2. Atomically increment TDCS.DIRTY_COUNT.

Completion Status Codes

Table 6.45: TDH.EXPORT.UNBLOCKW Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
TDX_EPT_WALK_FAILED	

Completion Status Code	Description
TDX_NOT_WRITE_BLOCKED	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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_TD_NOT_INITIALIZED	
TDX_TLB_TRACKING_NOT_DONE	

6.3.12. NEW: 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 6.46: TDH.IMPORT.ABORT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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		

Table 6.47: TDH.IMPORT.ABORT Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
AVX, AVX2 and AVX512 state	May be reset to the architectural INIT 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.

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 6.48: TDH.IMPORT.ABORT Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	R8	HPA	Memory to use for MBMD	MBMD	RW	Shared	128B	None	None	None
Explicit	R10	N/A	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 lower than TDCS.NUM_MIGS.
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 1 as the direction bit, 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 6.49: TDH.IMPORT.ABORT Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
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_TD_NOT_INITIALIZED	

6.3.13. NEW: TDH.IMPORT.COMMIT Leaf

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

Table 6.50: TDH.IMPORT.COMMIT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	HPA of the source TD's TDR page (HKID bits must be 0)		

Table 6.51: TDH.IMPORT.COMMIT Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
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.

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 6.52: TDH.IMPORT.COMMIT Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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 6.53: TDH.IMPORT.COMMIT Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
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_TD_NOT_INITIALIZED	

6.3.14. NEW: TDH.IMPORT.END Leaf

End an import session.

Table 6.54: TDH.IMPORT.END Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	HPA of the source TD's TDR page (HKID bits must be 0)		

Table 6.55: TDH.IMPORT.END Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
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).

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 6.56: TDH.IMPORT.END Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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.

If successful, the function does the following:

6. Set TDCS.OP_STATE to RUNNABLE.

Completion Status Codes

Table 6.57: TDH.IMPORT.END Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
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_TD_NOT_INITIALIZED	

6.3.15. NEW: 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 6.58: TDH.IMPORT.MEM Input Operands Definition

<table>
<tr><th>Operand</th><th colspan="3">Description</th></tr>
<tr><td>RAX</td><td colspan="3">SEAMCALL instruction leaf number and version, see 6.3.1</td></tr>
<tr><td></td><td>Bits</td><td>Field</td><td>Description</td></tr>
<tr><td></td><td>15:0</td><td>Leaf Number</td><td>Selects the SEAMCALL interface function</td></tr>
<tr><td></td><td>23:16</td><td>Version Number</td><td>Selects the SEAMCALL interface function version</td></tr>
<tr><td></td><td>63:24</td><td>Reserved</td><td>Must be 0</td></tr>
<tr><td>RCX</td><td colspan="3">GPA_LIST_INFO: HPA of a GPA list page in shared memory, and first and last entries to process, as defined in 4.10.2</td></tr>
<tr><td>RDX</td><td colspan="3">HPA of the destination TD’s TDR page (HKID bits must be 0)</td></tr>
<tr><td>R8</td><td colspan="3">HPA and size of memory of an MBMD structure in memory:</td></tr>
<tr><td></td><td>Bits</td><td>Name</td><td>Description</td></tr>
<tr><td></td><td>51:0</td><td>HPA</td><td>Bits 51:0 of the host physical address (including HKID bits)</td></tr>
<tr><td></td><td>63:52</td><td>Size</td><td>Size of the memory buffer containing MBMD, in bytes</td></tr>
<tr><td>R9</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 4.10.3.
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>R10</td><td colspan="3">Migration stream and resume flag:</td></tr>
<tr><td></td><td>Bits</td><td>Name</td><td>Description</td></tr>
<tr><td></td><td>15:0</td><td>MIGS_INDEX</td><td>Migration stream index</td></tr>
<tr><td></td><td>62:16</td><td>RESERVED</td><td>Reserved: must be 0</td></tr>
<tr><td></td><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 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 4.10.3.</td></tr>
<tr><td>R12</td><td colspan="3">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 4.10.3.</td></tr>
<tr><td>R13</td><td colspan="3">If in-place import is requested for 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, 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 4.8.6. For pages imported for the first-time in the current import session, or for the first-time after a previous import cancellation, the corresponding pages pointed the list become the destination TD pages.</td></tr>
</table>

Table 6.59: TDH.IMPORT.MEM Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
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
AVX, AVX2 and AVX512 state	May be reset to the architectural INIT 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, 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 least by requesting no operation for specific entries. The GPA list format is described in 4.10.2.

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.

Interruptibility: TDH.IMPORT.MEM is interruptible. If a pending interrupt is detected during operation, TDH.IMPORT.MEM returns with a TDX_INTERRUPED_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.

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 6.60: TDH.IMPORT.MEM Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	RW	Opaque	4KB	Shared	Shared	Shared
Explicit	R8	HPA	MBMD	MBMD	R	Shared	128B	None	None	None
Explicit	R9	HPA	Migration buffer list	PAGE_LIST	R	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	GPA	TD private pages (via GPA list)	Block	None	Private	4KB	None	None	None
Explicit	N/A	HPA	Migration buffer pages (via page list)	Blob	RW	Shared	4KB	None	None	None
Explicit	N/A	HPA	MAC pages (via page list)	Blob	RW	Shared	4KB	None	None	None
Explicit	N/A	HPA	Destination pages (via page list)	Blob	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	N/A	Migration context	N/A	RW	Opaque	N/A	None	N/A	N/A
Implicit	N/A	GPA	Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive	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. If import is in the in-order phase, check that the migration stream index is lower than TDCS.NUM_IN_ORDER_MIGS. Else, check that 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 0 as the direction bit, the stream index and the stream context's IV_COUNTER.
   9.3. Accumulate page MAC based on the GPA list entry.
   9.4. If no operation is requested:
      9.4.1. Check that the calculated MAC value is equal to the provided page MAC value.

      If passed:

      9.4.2. Mark the corresponding new page list entry (if available) as invalid and continue to the next GPA list entry.
   9.5. Walk the SEPT based on the GPA and level operands and find the leaf entry for the page.
   9.6. Check that the SEPT entry state is allowed for page import.
   9.7. If import is in the out-of-order phase, check that the requested operation in first-time import.
   9.8. 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.
   9.9. If the requested operation is first-time migrate:
      9.9.1. Check that the SEPT entry state is either FREE or REMOVED.
      9.9.2. If the SEPT entry state is REMOVED, check that the has not been removed in the current migration epoch.

      If passed:

      9.9.3. 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.
      9.9.4. Else, check that the new page list entry is a valid shared HPA.
      9.9.5. 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.
      9.9.6. Check that the calculated MAC value is equal to the provided page MAC value.

      If passed:

      9.9.7. Update the new TD page PAMT entry; record the current migration epoch value in PAMT.BEPOCH.
      9.9.8. Update the SEPT entry.
   9.10. Else, if the requested operation is re-migrate:
      9.10.1. Check that the SEPT entry state is either MAPPED or PENDING.
      9.10.2. Using the page's PAMT.BEPOCH, check that the page has not been imported in the current migration epoch.

If passed:

9.10.3. Record the current migration epoch value in PAMT.BEPOCH.
9.10.4. 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.
9.10.5. Check that the calculated MAC value is equal to the provided page MAC value.

If passed:

9.10.6. Update the SEPT entry.
9.11. Else, if the requested operation is migration cancel:
9.11.1. Check that the SEPT entry state indicates that the page has been exported.
9.11.2. Calculate MAC over the GPA list entry and check that the value is equal to the provided page MAC value.
9.11.3. Using the page's PAMT.BEPOCH, check that the page has not been imported in the current migration epoch.

If passed:

9.11.4. Update the SEPT entry; set the state to REMOVED and record the current migration epoch in the HPA.
9.12. If this is not the last round and there is a pending interrupt:
9.12.1. Save intermediate state in the migration stream context.
9.12.2. Terminate TDH.EXPORT.MEM with a TDX_INTERRUPTED_RESUMABLE status.
9.13. 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 6.61: TDH.IMPORT.MEM Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	
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_TD_NOT_INITIALIZED	

6.3.16. NEW: 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.

Table 6.62: TDH.IMPORT.STATE.IMMUTABLE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	HPA of Destination TD 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
R9	PAGE_LIST_INFO: migration buffers list information – see 0		
R10	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 6.63: TDH.IMPORT.STATE.IMMUTABLE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
RCX	Extended error information 1 In case of an error related to non-memory state field import, RCX contains the offending field identifier.
RDX	Extended error information 2 Reserved – set to 0.
AVX, AVX2 and AVX512 state	May be reset to the architectural INIT 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 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.

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.

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.

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 6.64: TDH.IMPORT.STATE.IMMUTABLE Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	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	N/A	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

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. 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. Check that a valid migration 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.

   If passed:

   9.2. Initialize the migration context in TDCS:
      9.2.1. Copy the migration key to a working migration key that will be used throughout the import session.
      9.2.2. Copy the MBMD's NUM_IN_ORDER_MIGS to TDCS
      9.2.3. Set all migration streams' INITIALIZED flag to 0 and ENABLED flags to 1.
   9.3. Initialize the current migration stream.
   9.4. Copy the MBMD into the migration context.
   9.5. Check the MBMD fields.

   If passed:

   9.6. 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.
   9.7. 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 6.2.1.3. 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. Initialize TDR and TDCS fields that need to be initialized at the beginning 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. Set TDCS.TOTAL_MB to 1.
      11.4.7. Set TDCS.OP_STATE to MEMORY_IMPORT.
      11.4.8. 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 6.65: TDH.IMPORT.STATE.IMMUTABLE Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	

Completion Status Code	Description
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_TD_NOT_INITIALIZED	

6.3.17. NEW: TDH.IMPORT.STATE.TD Leaf

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

Table 6.66: TDH.IMPORT.STATE.TD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	HPA of Destination TD 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
R9	PAGE_LIST_INFO: migration buffers list information – see 0		
R10	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 6.67: TDH.IMPORT.STATE.TD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
RCX	Extended error information 1 In case of an error related to non-memory state field import, as indicated by RAX, RCX contains the offending field identifier.
RDX	Extended error information 2 Reserved – set to 0.
AVX, AVX2 and AVX512 state	May be reset to the architectural INIT 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.

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.

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

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.

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 6.68: TDH.IMPORT.STATE.VP Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	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	N/A	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

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 0 as the direction bit, 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 6.2.1.3. 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 6.69: TDH.IMPORT.STATE.TD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	
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_TD_NOT_INITIALIZED	

6.3.18. NEW: TDH.IMPORT.STATE.VP Leaf

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

Table 6.70: TDH.IMPORT.STATE.VP Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	HPA of the source TD VCPU's TDVPR 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
R9	PAGE_LIST_INFO: migration buffers list information – see 0		
R10	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 6.71: TDH.IMPORT.STATE.VP Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
RCX	Extended error information 1 In case of an error related to non-memory state field import, as indicated by RAX, RCX contains the offending field identifier.
RDX	Extended error information 2 Reserved – set to 0.
AVX, AVX2 and AVX512 state	May be reset to the architectural INIT 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.

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.

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

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.

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 Spec].

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 6.72: TDH.IMPORT.STATE.VP Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	N/A	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
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_IN_ORDER_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. 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):
    10.1. Copy the MBMD into the migration context.
    10.2. Check the MBMD fields.

    If passed:

    10.3. 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.
    10.4. Accumulate MAC based on the MAC'ed fields of MBMD.
    10.5. Atomically increment the TD's VCPU counter (TDCS.NUM_VCPUS), and check that maximum number of VCPUs (TDCS.MAX_VCPUS) has not been exceeded.
11. Else (this is a resumption of a previously interrupted TDH.IMPORT.STATE.VP):
    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 are the same as in the interruption state.

If passed:

12. Repeat importing 4KB pages until all TD-scope state is imported or until a pending interrupt is detected:
    12.1. Get the 4KB next page HPA from it from the page list.
    12.2. 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.
    12.3. Parse the metadata list and write the control structure fields using the algorithm described in 6.2.1.3. Check each TDVPS field for compatibility.

    If passed:

    12.4. If all metadata lists have been imported:
        12.4.1. Check that the accumulated MAC value is equal to the saved MBMD's MAC value.
        12.4.2. Check that all TDR and TDCS fields required to be imported by TDH.IMPORT.STATE.VP have indeed been imported.
        12.4.3. Initialize TDVPS fields that need to be initialized at the end of the import session.
        12.4.4. Mark the migration stream context's interrupted state as invalid.
        12.4.5. Increment the migration stream context's EXPECTED_MB_COUNTER.
        12.4.6. Increment TDCS.TOTAL_MB.
        12.4.7. Terminate TDH.IMPORT.STATE.VP 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.IMPORT.STATE.VP with a TDX_INTERRUPTED_RESUMABLE status.

Completion Status Codes

Table 6.73: TDH.IMPORT.STATE.VP Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	Operation is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	

Completion Status Code	Description
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_INITIALIZED	

6.3.19. NEW: 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 6.74: TDH.IMPORT.TRACK Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 6.75: TDH.IMPORT.TRACK Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code, see 6.3.1
AVX, AVX2 and AVX512 state	May be reset to the architectural INIT 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.

A failure marks 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.

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 6.76: TDH.IMPORT.TRACK Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	N/A	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 0 as the direction bit, 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. Set the stream context's EXPECTED_MB_COUNTER to 1.
15. Increment TDCS.TOTAL_MB.
16. Set TDCS.MIG_EPOCH to the MIG_EPOCH value provided in the MBMD.
17. If the MIG_EPOCH value provided in the MBMD is 0xFFFFFFFF, indicating the start of out-of-order phase:
    17.1. Start the out-of-order import phase: set TDCS.OP_STATE to POST_IMPORT.

Completion Status Codes

Table 6.77: TDH.IMPORT.TRACK Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
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_TD_NOT_INITIALIZED	

6.3.20. UPDATED: 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 6.78: TDH.MEM.PAGE.ADD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the EPT entry that will map the new page – see 4.5.1: 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 6.79: TDH.MEM.PAGE.ADD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 4.5.2 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 4.5.2 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.

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 6.80: TDH.MEM.PAGE.ADD Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA	TD private page (GPA)[2]	Blob	RW	Private	4KB	N/A	N/A	N/A
Explicit	RDX	HPA	TDR page	Blob	RW	Opaque	4KB	Exclusive	Shared	Shared
Explicit	R8	HPA	TD private page (HPA)[2]	Blob	RW	Private	4KB	Exclusive	Shared	Shared
Explicit	R9	HPA	Source page	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	GPA	Secure EPT tree	N/A	RW	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

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.

If the Secure EPT entry is marked as FREE, the function does the following:

7. Copy the source image to the target TD page using the TD's ephemeral private HKID, and direct write (MOVDIR64B).
8. Update the parent Secure EPT entry with the target page HPA and MAPPED state.
9. 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.

[2] RCX and R8 denote the same TD private page operand, using HPA and GPA respectively

10. Increment TDR.CHLDCNT.
11. Update the PAMT entry with the PT_REG page type and the TDR physical address as the OWNER.

Completion Status Codes

Table 6.81: TDH.MEM.PAGE.ADD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_NOT_FREE	
TDX_EPT_WALK_FAILED	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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_FINALIZED	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_INITIALIZED	

6.3.21. UPDATED: TDH.MEM.PAGE.AUG Leaf

Dynamically add a 4KB or a 2MB private page to an initialized TD, mapped to the specified GPAs.

Table 6.82: TDH.MEM.PAGE.AUG Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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
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 6.83: TDH.MEM.PAGE.AUG Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 4.5.2 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 4.5.2 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.

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 6.84: TDH.MEM.PAGE.AUG Memory Operands Information Definition

Explicit/ Implicit	Register	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA	TD private page (GPA)[3]	Blob	None	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	TD private page (HPA)[3]	Blob	None	Private	$2^{12+9*Level}$ Bytes	Exclusive	Shared[4]	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	Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	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.

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. Update the PAMT entry with the PT_REG page type and the TDR physical address as the OWNER.

Completion Status Codes

Table 6.85: TDH.MEM.PAGE.AUG Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_NOT_FREE	
TDX_EPT_WALK_FAILED	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	

[3] RCX and R8 denote the same TD private page operand, using HPA and GPA respectively

[4] Applicable for 4KB pages only

Completion Status Code	Description
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_TD_NOT_FINALIZED	
TDX_TD_NOT_INITIALIZED	

6.3.22. UPDATED: TDH.MEM.PAGE.DEMOTE Leaf

Split a large private TD page (2MB or 1GB) into 512 small pages (4KB or 2MB, respectively).

Table 6.86: TDH.MEM.PAGE.DEMOTE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the Secure EPT entry that maps the large page to be split: either 1 (2MB) or 2 (1GB) – see 4.5.1
	11:3	Reserved	Reserved: must be 0
	51:12	GPA	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
	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 new Secure EPT page to be added to the TD (HKID bits must be 0)		

Table 6.87: TDH.MEM.PAGE.DEMOTE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 4.5.2 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 4.5.2 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.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.

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 6.88: TDH.MEM.PAGE.DEMOTE Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. Type	Resource Name	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA and Level	TD private page to split	Blob	None	Private	$2^{12+9*level}$ bytes	Exclusive	None	None
Explicit	RDX	HPA	TDR page	TDR	RW	Opaque	4KB	Shared	Shared	Shared
Explicit	R8	HPA	New Secure EPT page	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	Secure EPT Tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	N/A	N/A

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 4.5.1 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 large TD private page to be demoted.
7. Check the page is blocked (its parent Secure EPT entry is a leaf entry, and its state is BLOCKED or PENDING_BLOCKED).
8. Check that TLB tracking has been done, based on the large TD private page's PAMT.BEPOCH.

If successful, the function does the following:

9. Split the large TD private page PAMT entry into 512 PAMT entries at the lower level:
   9.1. Set the parent PAMT_2M or PAMT_1G entry state to PT_NDA.
   9.2. Set the 512 child PAMT4K or PAMT_2M entries respectively to PT_REG.
10. Initialize the new Secure EPT page's 512 entries to MAPPED state pointing to the 512 consecutive small pages above. Use the TD's ephemeral private HKID and direct write (MOVDIR64B).
11. Atomically set the demoted Secure EPT entry to MAPPED (if it was BLOCKED) or PENDING (if it was PENDING_BLOCKED) non-leaf entry pointing to the new Secure EPT page.
12. Atomically increment TDR.CHLDCNT by 1.

12.1. Note that CHLDCNT counts the number of 4KB pages. The change is due only to the addition of the new Secure EPT page.

13. Update the PAMT entry of the new Secure-EPT page with the PT_EPT page type and the TDR physical address as the OWNER.

Completion Status Codes

Table 6.89: TDH.MEM.PAGE.DEMOTE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_NOT_LEAF	
TDX_EPT_WALK_FAILED	
TDX_GPA_RANGE_NOT_BLOCKED	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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	
TDX_TD_NOT_INITIALIZED	
TDX_TLB_TRACKING_NOT_DONE	

6.3.23. UPDATED: TDH.MEM.PAGE.PROMOTE Leaf

Merge 512 consecutive small private TD pages (4KB or 2MB) into one large page (2MB or 1GB, respectively).

Table 6.90: TDH.MEM.PAGE.PROMOTE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 4.5.1)
	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	Host physical address of the parent TDR page (HKID bits must be 0)		

Table 6.91: TDH.MEM.PAGE.PROMOTE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 4.5.2 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 4.5.2 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.PROMOTE merges 512 private pages, which are consecutive both in the HPA space and in the GPA space. It removes the Secure EPT leaf page that formerly mapped those pages.

All merged private pages must have the same Secure EPT attributes and state, which must be either MAPPED or PENDING.

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 6.92: TDH.MEM.PAGE.PROMOTE Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA and Level	Removed 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	HPA	Merged HPA range	Blob	None	Private	N/A	Exclusive	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	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT Tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Large page Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	N/A	N/A
Implicit	N/A	GPA	Small pages Secure EPT entries	SEPT Entry	RW	Private	N/A	Exclusive	N/A	N/A

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 (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is either INITIALIZED or RUNNING).
5. The specified merged page level is either 1 (2MB) or 2 (1GB) – see 4.5.1 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. Check the Secure EPT entry:
   7.1. It must be a non-leaf entry.
   7.2. It must be blocked (BLOCKED) .
8. 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.

9. Check that TLB tracking has been done, based on the above Secure EPT page's PAMT.BEPOCH.
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 MAPPED
    10.3. Have contiguous HPA mapping aligned to the promoted range size.

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:

11. Merge the corresponding 512 physical pages into a single larger physical page:
    11.1. Set the small page (PAMT_4K or PAMT_2M) entries state to PT_NDA.
    11.2. Set the parent (PAMT_2M or PAMT_1G respectively) entry to PT_REG.
12. 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.
13. Remove the Secure EPT page that previously mapped the 512 physical pages:
    13.1. Atomically decrement TDR.CHLDCNT by 1.
        13.1.1. Note that CHLDCNT counts the number of 4KB pages. The change is due only to the removal of the Secure EPT page.
    13.2. Update the PAMT entry of the removed Secure EPT page to PT_NDA.

Completion Status Codes

Table 6.93: TDH.MEM.PAGE.PROMOTE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_FREE	
TDX_EPT_ENTRY_LEAF	
TDX_EPT_INVALID_PROMOTE_CONDITIONS	
TDX_EPT_WALK_FAILED	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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_TD_NOT_INITIALIZED	
TDX_TLB_TRACKING_NOT_DONE	

6.3.24. UPDATED: TDH.MEM.PAGE.RELOCATE Leaf

Relocate a 4KB mapped page from its current host physical address to another.

Table 6.94: TDH.MEM.PAGE.RELOCATE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 4.5.1).
	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 6.95: TDH.MEM.PAGE.RELOCATE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 4.5.2 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 4.5.2 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.

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 6.96: TDH.MEM.PAGE.RELOCATE Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA and Level	TD private page	Blob	R	Private	4KB	Exclusive	None	None
Explicit	RDX	HPA	TDR page	TDR	RW	Opaque	4KB	Shared	Shared	Shared
Explicit	R8	HPA	Target physical page	Blob	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	Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	N/A	N/A

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).
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. Check the Secure EPT entry is a blocked (BLOCKED or PENDING_BLOCKED) leaf entry.
8. Check that the currently mapped HPA is different than the target HPA.
9. Check that TLB tracking was done.

If successful, the function does the following:

10. If the page state is BLOCKED, copy the currently mapped page content to the target page, using the TD's ephemeral private HKID and direct writes (MOVDIR64B).
11. Free the currently mapped HPA by setting its PAMT.PT to PT_NDA.
12. Update the target page's PAMT entry with the PT_REG page type and the TDR physical address as the OWNER.
13. Update the Secure EPT entry with the target page HPA. Set its state to MAPPED or PENDING depending on whether its previous state was BLOCKED or PENDING_BLOCKED, respectively.

Completion Status Codes

Table 6.97: 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_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_FINALIZED	
TDX_TD_NOT_INITIALIZED	
TDX_TLB_TRACKING_NOT_DONE	

6.3.25. UPDATED: TDH.MEM.PAGE.REMOVE Leaf

Remove a GPA-mapped 4KB, 2MB or 1GB private page from a TD.

Table 6.98: TDH.MEM.PAGE.REMOVE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 4.5.1.
	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 6.99: TDH.MEM.PAGE.REMOVE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 4.5.2 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 4.5.2 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. On successful operation, it marks the physical page as free in PAMT.

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 6.100: TDH.MEM.PAGE.REMOVE Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	GPA and Level	TD private page	Blob	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	Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	N/A	N/A

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.

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.
5. The specified level is either 0 (4KB), 1 (2MB) or 2 (1GB) – see 4.5.1 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 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 page's parent Secure EPT entry is a blocked leaf entry (BLOCKED or PENDING_BLOCKED).
   7.2. Check that TLB tracking was done.

If successful, the function does the following:

8. Atomically decrement TDR.CHLDCNT by 1, 512 or 512^2 depending on the removed TD private page size (4KB, 2MB or 1GB, respectively).
9. Free the physical page:
   9.1. If the level is 0 (4KB), set the PAMT entry of the removed TD private page to PT_NDA.
   9.2. Else (levels 1 or 2, 2MB or 1GB respectively), set the PAMT entry of the removed TD private page to PT_NDA.
10. Set the parent Secure EPT entry to FREE.

Completion Status Codes

Table 6.101: TDH.MEM.PAGE.REMOVE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_WALK_FAILED	
TDX_GPA_RANGE_NOT_BLOCKED	
TDX_OPERAND_ADDR_RANGE_ERROR	

Completion Status Code	Description
TDX_OPERAND_BUSY	
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_TD_NOT_FINALIZED	
TDX_TD_NOT_INITIALIZED	
TDX_TLB_TRACKING_NOT_DONE	

6.3.26. UPDATED: 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 6.102: TDH.MEM.RANGE.BLOCK Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the Secure EPT entry that maps the GPA range to be blocked – see 4.5.1 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 6.103: TDH.MEM.RANGE.BLOCK Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 4.5.2 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 4.5.2 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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.104: TDH.MEM.RANGE.BLOCK Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	Secure EPT tree	N/A	RW	Private	N/A	Exclusive	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	Transaction	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 4.5.1 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 MAPPED or PENDING).

If passed:

8. Block the Secure EPT entry. Use an atomic operation (LOCK CMPXCHG) to check that the Secure EPT entry has not change and to set its state to BLOCKED (if it was MAPPED) or PENDING_BLOCKED (if it was PENDING).

If passed:

9. 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 6.105: TDH.MEM.RANGE.BLOCK Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_FREE	
TDX_EPT_WALK_FAILED	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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_TD_NOT_INITIALIZED	

6.3.27. UPDATED: 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 6.106: TDH.MEM.RANGE.UNBLOCK Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the Secure EPT entry that maps the GPA range to be unblocked – see 4.5.1 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)		

5

Table 6.107: TDH.MEM.RANGE.UNBLOCK Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 4.5.2 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 4.5.2 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).

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 6.108: TDH.MEM.RANGE.UNBLOCK Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	Secure EPT tree	N/A	RW	Private	N/A	Exclusive	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	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 4.5.1 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 (BLOCKED or PENDING_BLOCKED).
8. Check that TLB tracking was done.

If successful, the function does the following:

9. Unblock the Secure EPT entry. Atomically set its state to MAPPED (if it was BLOCKED) or PENDING (if it was PENDING_BLOCKED).

Completion Status Codes

Table 6.109: 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_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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_TD_NOT_INITIALIZED	
TDX_TLB_TRACKING_NOT_DONE	

6.3.28. TDH.MEM.RD Leaf

Read a 64b chunk from a debuggable guest TD private memory.

Table 6.110: TDH.MEM.RD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 6.111: TDH.MEM.RD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 4.5.2 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 4.5.2 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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.112: TDH.MEM.RD Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	GPA	Secure EPT tree	N/A	RW	Private	N/A	Shared	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 6.113: TDH.MEM.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TO BE COMPLETED	
TDX_SUCCESS	

6.3.29. UPDATED: TDH.MEM.SEPT.ADD Leaf

Add and map a 4KB Secure EPT page to a TD.

Table 6.114: TDH.MEM.SEPT.ADD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the non-leaf Secure EPT entry that will map the new Secure EPT page – see 4.5.1 Level must 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 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
	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 new Secure EPT page to be added to the TD (HKID bits must be 0)		

Table 6.115: TDH.MEM.SEPT.ADD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 4.5.2 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 4.5.2 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.SEPT.ADD adds a 4KB Secure EPT page to a TD and maps it to the provided GPA. It initializes the page to hold 512 free entries using the TD's ephemeral private key.

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 6.116: TDH.MEM.SEPT.ADD Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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)[5]	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)[5]	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	Secure EPT tree	N/A	RW	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	Exclusive	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 (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is either INITIALIZED or RUNNING).
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 4.5.1 for a definition of EPT level.
6. The target page metadata in PAMT must be correct (PT must be PT_NDA).

[5] RCX and R8 denote the same Secure EPT page operand, using HPA and GPA respectively

If successful, the function does the following:

7. Walk the Secure EPT based on the GPA operand, and find the parent EPT entry for the new Secure EPT page.

If the Secure EPT entry is marked as FREE:

8. Initialize the new Secure EPT page to 0, indicating 512 entries in the FREE state, using the TD's ephemeral private HKID and direct writes (MOVDIR64B).
9. Update the parent Secure EPT entry with the new Secure EPT page HPA and MAPPED state.
10. Increment TDR.CHLDCNT.
11. Update the new Secure EPT page's PAMT entry with the PT_EPT page type and the TDR physical address as the OWNER.

Completion Status Codes

Table 6.117: TDH.MEM.SEPT.ADD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_NOT_FREE	
TDX_EPT_WALK_FAILED	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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_TD_NOT_INITIALIZED	

6.3.30. UPDATED: TDH.MEM.SEPT.RD Leaf

Read a Secure EPT entry.

Table 6.118: TDH.MEM.SEPT.RD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the Secure EPT entry to read – see 4.5.1 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	Host physical address of the parent TDR page (HKID bits must be 0)		

Table 6.119: TDH.MEM.SEPT.RD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RCX	Secure EPT entry architectural content – see 4.5.2 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.

Operand	Description
RDX	Secure EPT entry level and state – see 4.5.2 • 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.
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.RD reads a Secure EPT entry.

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 6.120: TDH.MEM.SEPT.RD Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	Secure EPT Tree	N/A	R	Private	N/A	Shared	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	SEPT Entry	RW	Private	N/A	None	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 4.5.1 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.
7. Read the Secure EPT entry contents.

Completion Status Codes

Table 6.121: TDH.MEM.SEPT.RD 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_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MEM.SEPT.RD is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_INITIALIZED	

6.3.31. UPDATED: TDH.MEM.SEPT.REMOVE Leaf

Remove an empty 4KB Secure EPT page from a TD.

Table 6.122: TDH.MEM.SEPT.REMOVE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 4.5.1 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 6.123: TDH.MEM.SEPT.REMOVE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RCX	Extended error information part 1 In case of EPT walk error, Secure EPT entry architectural content where the error was detected – see 4.5.2 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 4.5.2 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.SEPT.REMOVE removes an empty Secure EPT page, with all 512 marked as FREE, from the TD's Secure EPT tree. On successful operation, it marks the 4KB physical page as free in PAMT.

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 6.124: TDH.MEM.SEPT.REMOVE Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	Secure EPT Tree	N/A	RW	Private	N/A	Exclusive	N/A	N/A
Implicit	N/A	GPA	Secure EPT entry	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 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 a non-leaf EPT entry (i.e., 1 to 3 for 4-level EPT or 1 to 4 for 5-level EPT) – see 4.5.1 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 to be removed.
7. Check the page's parent Secure EPT entry is a blocked (BLOCKED) non-leaf entry.
8. Check that TLB tracking was done.
9. Scan the Secure EPT page content and check all 512 entries are FREE.

If successful, the function does the following:

10. Atomically decrement TDR.CHLDCNT.
11. Set the PAMT entry of the removed Secure EPT page to PT_NDA.
12. Set the parent Secure EPT entry to FREE.

Completion Status Codes

Table 6.125: TDH.MEM.SEPT.REMOVE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_NOT_FREE	
TDX_EPT_WALK_FAILED	
TDX_GPA_RANGE_NOT_BLOCKED	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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_TD_NOT_INITIALIZED	
TDX_TLB_TRACKING_NOT_DONE	

6.3.32. UPDATED: TDH.MEM.TRACK Leaf

Increment the TD's TLB epoch counter.

Table 6.126: TDH.MEM.TRACK Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of the parent TDR page (HKID bits must be 0)		

Table 6.127: TDH.MEM.TRACK Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.128: TDH.MEM.TRACK Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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 and finalized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is RUNNING).

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 6.129: TDH.MEM.TRACK Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
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_TD_NOT_FINALIZED	
TDX_TD_NOT_INITIALIZED	

6.3.33. TDH.MEM.WR Leaf

Write a 64b chunk from a debuggable guest TD private memory.

Table 6.130: TDH.MEM.WR Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 6.131: TDH.MEM.WR Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RCX	Secure EPT entry architectural content – see 4.5.2 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 4.5.2 • 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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.132: TDH.MEM.RD Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	GPA	Secure EPT tree	N/A	RW	Private	N/A	Shared	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 6.133: TDH.MEM.WR Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TO BE COMPLETED	
TDX_SUCCESS	

6.3.34. NEW: TDH.MIG.STREAM.CREATE Leaf

Create a Migration Stream and its MIGSC control structure.

Table 6.134: TDH.MIG.STREAM.CREATE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 6.135: TDH.MIG.STREAM.CREATE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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.

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 6.136: TDH.MIG.STREAM.CREATE Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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
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	Shared	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. Initialize the MIGSC page contents using direct write (MOVDIR64B).
10. 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.
11. Atomically increment TDR.CHLDCNT.
12. Initialize the MIGSC page metadata in PAMT (Set PT to PT_TDCX, OWNER to the TDR HPA).

Completion Status Codes

Table 6.137: TDH.MIG.STREAM.CREATE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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_FINALIZED	
TDX_TD_KEYS_NOT_CONFIGURED	

6.3.35. UPDATED: TDH.MNG.ADDCX Leaf

Add a TDCX page to a guest TD.

Table 6.138: TDH.MNG.ADDCX Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 6.139: TDH.MNG.ADDCX Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 TDCX page, which is a child of the specified TDR.

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 6.140: TDH.MNG.ADDCX Memory Operands Information Definition

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

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 must not have been initialized (TDR.INIT is FALSE).~~
4. The number of TDCX pages (TDR.NUM_TDCX) is smaller than the required number.
5. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
6. The new TDCX page metadata in PAMT must be correct (PT must be PT_NDA).

If successful, the function does the following:

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, mark the TD as uninitialized (set TDCS.OP_STATE to UNINITIALIZED).

Completion Status Codes

Table 6.141: TDH.MNG.ADDCX Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MNG.ADDCX is successful
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_INITIALIZED	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCX_NUM_INCORRECT	

6.3.36. TDH.MNG.CREATE Leaf

Create a new guest TD and its TDR root page.

Table 6.142: TDH.MNG.CREATE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 6.143: TDH.MNG.CREATE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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.

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 6.144: TDH.MNG.CREATE Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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 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. Zero out the TDR page contents using direct write (MOVDIR64B).

5. Initialize the key management fields.
6. Initialize the state variables.
7. Initialize the TD management fields.
8. Mark the KOT entry for the specified HKID as HKID_ASSIGNED.
9. Initialize the TDR page metadata in PAMT.

Completion Status Codes

Table 6.145: 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	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MNG.CREATE is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	

6.3.37. UPDATED: TDH.MNG.INIT Leaf

Initialize TD-scope control structures TDR and TDCS.

Table 6.146: TDH.MNG.INIT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of a TDR page (HKID bits must be 0)		
RDX	The physical address (including HKID bits) of an input TD_PARAMS_STRUCT		

Table 6.147: TDH.MNG.INIT Output Operands Definition

Operand	Description		
RAX	SEAMCALL instruction return code – see 6.3.1		
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.

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 6.148: TDH.MNG.INIT Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	RDX	HPA	TD Parameters	TD_PARAMS	R	Shared	1024B	None	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

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 must not have been initialized (TDR.INIT is FALSE).~~
4. The TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
5. 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).

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).

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 6.149: TDH.MNG.INIT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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_INITIALIZED	
TDX_TD_KEYS_NOT_CONFIGURED	

Completion Status Code	Description
TDX_TDCX_NUM_INCORRECT	

6.3.38. TDH.MNG.KEY.CONFIG Leaf

Configure the TD ephemeral private key on a single package.

Table 6.150: TDH.MNG.KEY.CONFIG Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of a TDR page (HKID bits must be 0)		

Table 6.151: TDH.MNG.KEY.CONFIG Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.152: TDH.MNG.KEY.CONFIG Operands Information

Explicit/ Implicit	Reg.	Addr. 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 6.153: TDH.MNG.KEY.CONFIG Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_KEY_CONFIGURED	
TDX_KEY_GENERATION_FAILED	
TDX_LIFECYCLE_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MNG.KEY.CONFIG is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	

6.3.39. TDH.MNG.KEY.FREEID Leaf

End the platform cache flush sequence, and mark applicable HKIDs in KOT as free.

Table 6.154: TDH.MNG.KEY.FREEID Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of a TDR page (HKID bits must be 0)		

Table 6.155: TDH.MNG.KEY.FREEID Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 on all the required packages. 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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.156: TDH.MNG.KEY.FREEID Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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:
   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. 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 6.157: 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	
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	

6.3.40. TDH.MNG.KEY.RECLAIMID Leaf

This function is provided for backward compatibility.

Table 6.158: TDH.MNG.KEY.RECLAIMID Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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

Table 6.159: TDH.MNG.KEY.RECLAIMID Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 6.160: TDH.MNG.KEY.RECLAIMID Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_SUCCESS	TDH.MNG.KEY.RECLAIMID is successful.

6.3.41. UPDATED: TDH.MNG.RD Leaf

Read a TD-scope metadata field (control structure field) of a TD.

Table 6.161: TDH.MNG.RD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of a TDR page (HKID bits must be 0)		
RDX	Field identifier – see 4.8 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 6.162: TDH.MNG.RD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RDX	For TDH.MNG.RD version 0, RDX is unmodified. For TDH.MNG.RD version 1 or higher, 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 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.RD reads a TD-scope metadata field (control structure field) of a 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 6.163: TDH.MNG.RD Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	R	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 6.2.1.1.

Completion Status Codes

Table 6.164: TDH.MNG.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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	
TDX_TD_NOT_INITIALIZED	

6.3.42. TDH.MNG.VPFLUSHDONE Leaf

Check that none of the TD's VCPUs are associated with an LP.

Table 6.165: TDH.MNG.VPFLUSHDONE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of a TDR page (HKID bits must be 0)		

Table 6.166: TDH.MNG.VPFLUSHDONE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.167: TDH.MNG.VPFLUSHDONE Operands Information

Explicit/ Implicit	Reg.	Addr. 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

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 6.168: TDH.MNG.VPFLUSHDONE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_FLUSHVP_NOT_DONE	
TDX_LIFECYCLE_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MNG.VPFLUSHDONE is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	

6.3.43. UPDATED: TDH.MNG.WR Leaf

Write a TD-scope metadata field (control structure field) of a TD.

Table 6.169: TDH.MNG.WR Input Operands Definitions

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of a TDR page (HKID bits must be 0)		
RDX	Field identifier – see 4.8 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 For TDH.MNG.WR version > 0, if the WRITE_MASK_VALID bit of the field identifier is 0, then the write mask is ignored.		

Table 6.170: TDH.MNG.WR Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 specific bits of the value (R8) are written as specified by the write mask (R9). Writing is subject to the field's writability.

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 6.171: TDH.MNG.WR Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Implicit	N/A	N/A	TDCS structure	TDCS	R	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).
5. The TD is debuggable (TDCS.ATTRIBUTES.DEBUG is 1)

If the above checks passed:

6. Write the control structure field and return its old value, using the algorithm described in 6.2.1.2.

Completion Status Codes

Table 6.172: TDH.MNG.WR Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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_TD_NOT_INITIALIZED	

6.3.44. UPDATED: TDH.MR.EXTEND Leaf

Extend the MRTD measurement register in the TDCS with the measurement of the indicated chunk of a TD page.

Table 6.173: TDH.MR.EXTEND Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 6.174: TDH.MR.EXTEND Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.175: TDH.MR.EXTEND Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
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 6.176: TDH.MR.EXTEND Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_EPT_ENTRY_NOT_PRESENT	
TDX_EPT_WALK_FAILED	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MR.EXTEND is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_FINALIZED	
TDX_TD_KEYS_NOT_CONFIGURED	

Completion Status Code	Description
TDX_TD_NOT_INITIALIZED	

6.3.45. UPDATED: TDH.MR.FINALIZE Leaf

TDH.MR.FINALIZE completes measurement of the initial TD contents and marks the TD as ready to run.

Table 6.177: TDH.MR.FINALIZE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of the parent TDR page (HKID bits must be 0)		

Table 6.178: TDH.MR.FINALIZE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.179: TDH.MR.FINALIZE Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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 6.180: TDH.MR.FINALIZE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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_FINALIZED	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_INITIALIZED	

6.3.46. TDH.PHYMEM.CACHE.WB Leaf

TDH.PHYMEM.CACHE.WB is an interruptible and restartable function to write back the cache hierarchy on a package or a core.

Table 6.181: TDH.PHYMEM.CACHE.WB Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 6.182: TDH.PHYMEM.CACHE.WB Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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.

- TDH.PHYMEM.CACHE.WB does not invalidate cache lines.
- The function is interruptible by external events and is restartable. In case it is interrupted by an external event, information is stored in an Intel TDX module internal table which allows the instruction to be restarted.
- The function operates on cache lines associated with any HKID.
- The function helps 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 == TDH.PHYMEM.CACHE.WB_START_CMD (0)).
- 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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.183: TDH.PHYMEM.CACHE.WB (Implicit) Operands Information

Explicit/ Implicit	Reg.	Addr. 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 and invalidated 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 package or core. 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 6.184: TDH.PHYMEM.CACHE.WB Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INTERRUPTED_RESUMABLE	
TDX_NO_HKID_READY_TO_WBCACHE	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDH.PHYMEM.CACHE.WB is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	

6.3.47. TDH.PHYMEM.PAGE.RDMD Leaf

Read the metadata of a page (or the metadata of the containing large page) in TDMR.

Table 6.185: TDH.PHYMEM.PAGE.RDMD Operands

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	A physical address of a 4KB page in TDMR (HKID bits must be 0)		

Table 6.186: TDH.PHYMEM.PAGE.RDMD Output Operands Definition

Operand	Description		
RAX	SEAMCALL instruction return code – see 6.3.1		
RCX	Page Type (PT):		
	Value	**Name**	**Description**
	0	PT_NDA	The physical page is **Not Directly Assigned** to the Intel TDX module.
	1	PT_RSVD	The physical page is reserved for non-TDX usage.
	3	PT_REG	The physical page holds TD private memory.
	4	PT_TDR	The physical page holds the TD Root (TDR) control structure.
	8:5		The physical page holds a TD control structure.
	Other		Reserved
	In case of an error, RCX returns 0.		
RDX	OWNER: the HPA of the TD's TDR control structure page (if applicable) In case of an error, as indicated by RAX, RDX returns 0.		
R8	**Bits**	**Name**	**Description**
	2:0	Size	Size of the containing 4KB, 2MB or 1GB page – see 4.4.1
	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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.187: TDH.PHYMEM.PAGE.RDMD Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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 6.188: TDH.PHYMEM.PAGE.RDMD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDH.PHYMEM.PAGE.RDMD is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	

6.3.48. 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 6.189: TDH.PHYMEM.PAGE.RECLAIM Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of a 4KB, 2MB or 1GB page to be reclaimed (HKID bits must be 0)		

5

Table 6.190: TDH.PHYMEM.PAGE.RECLAIM Output Operands Definition

Operand	Description		
RAX	SEAMCALL instruction return code – see 6.3.1		
RCX	Page Type (PT):		
	Value	**Name**	**Description**
	0	PT_NDA	The physical page is **Not Directly Assigned** to the Intel TDX module.
	1	PT_RSVD	The physical page is reserved for non-TDX usage.
	3	PT_REG	The physical page holds TD private memory.
	4	PT_TDR	The physical page holds the TD Root (TDR) control structure.
	8:5		The physical page holds a TD control structure.
	Other		Reserved
	In case of an error, as indicated by RAX, RCX returns 0.		
RDX	OWNER: the HPA of the TD's TDR control structure page (if applicable) In case of an error, as indicated by RAX, RDX returns 0.		
R8	**Bits**	**Name**	**Description**
	2:0	Size	Size of the containing 4KB, 2MB or 1GB page – see 4.4.1
	63:3	Reserved	Set to 0
	In case of an error, as indicated by RAX, R8 returns 0.		
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.

TDH.PHYMEM.PAGE.RECLAIM can reclaim pages only if the owner TD is in the TD_TEARDOWN state.

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 6.191: TDH.PHYMEM.PAGE.RECLAIM Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	Target page	Blob	RW	Opaque/ Private	4KB, 2MB or 1GB	Exclusive	Shared	Shared
Implicit	N/A	N/A	TDR page[6]	TDR	RW	Opaque	4KB	Shared	N/A	N/A

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. Check that the TD is in teardown state (TDR.LIFECYCLE_STATE is TD_TEARDOWN).
   2.3. Atomically decrement TDR.CHLDCNT.
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.
4. Update the PAMT entry of the reclaimed page to PT_NDA.
5. Return the page metadata (as they were before PAMT update above).

Completion Status Codes

Table 6.192: TDH.PHYMEM.PAGE.RECLAIM Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_LIFECYCLE_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	

[6] Except when TDR is the target page

Completion Status Code	Description
TDX_SUCCESS	TDH.PHYMEM.PAGE.RECLAIM is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_ASSOCIATED_PAGES_EXIST	

6.3.49. TDH.PHYMEM.PAGE.WBINVD Leaf

Write back and invalidate all cache lines associated with the specified memory page and HKID.

Table 6.193: TDH.PHYMEM.PAGE.WBINVD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	Physical address (including HKID bits) of a 4KB page in TDMR		

Table 6.194: TDH.PHYMEM.PAGE.WBINVD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.195: TDH.PHYMEM.PAGE.WBINVD Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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 6.196: TDH.PHYMEM.PAGE.WBINVD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.PHYMEM.PAGE.WBINVD is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	

6.3.50. NEW: TDH.SERVTD.BIND Leaf

Bind a service TD to a target TD.

Table 6.197: TDH.SERVTD.BIND Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 6.198: TDH.SERVTD.BIND Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 195:128 In case of an error, as indicated by RAX, R12 returns 0.
R13	TD_UUID bits 255:196 In case of an error, as indicated by RAX, R13 returns 0.
AVX, AVX2 and AVX512 state	May be reset to the architectural INIT 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.

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 6.199: TDH.SERVTD.BIND Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	RW	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	R	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 page metadata in PAMT must be correct (PT must be PT_TDR).
   2.2. The service TD is not in a FATAL state (TDR.FATAL is FALSE).
   2.3. The service TD keys are configured on the hardware (TDR.LIFECYCLE_STATE is TD_KEYS_CONFIGURED).
   2.4. The service TD's TDCS pages must have been allocated (TDR.NUM_TDCX is the required number).
   2.5. 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 or UNBOUND (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 and check that it is equal to the binding slot's SERVTD_INFO_HASH.
   9.4. If SERVTD_ATTR.INSTANCE_BINDING is set:
      9.4.1. Check that the SERVTD's TD_UUID is equal to the binding slot's SERVTD_UUID.
      9.4.2. Copy the calculated service TD's SERVTD_INFO_HASH to the binding slot's SERVTD_INFO_HASH.
   9.5. Else:
      9.5.1. Check that the service TD's SERVTD_INFO_HASH is equal to the binding slot's SERVTD_INFO_HASH.
      9.5.2. 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 6.200: TDH.SERVTD.BIND Completion Status Codes (Returned in RAX) Definition [TO BE EDITED]

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.SERVTD.BIND is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_INITIALIZED	

6.3.51. NEW: TDH.SERVTD.PREBIND Leaf

Pre-bind a service TD to a target TD.

Table 6.201: TDH.SERVTD.PREBIND Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 6.202: TDH.SERVTD.PREBIND Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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.

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 6.203: TDH.SERVTD.PREBIND Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	RW	Opaque	4KB	Shared	Shared	Shared
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

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
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 6.204: TDH.SERVTD.PREBIND Completion Status Codes (Returned in RAX) Definition [TO BE EDITED]

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.SERVTD.PREBIND is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_INITIALIZED	

6.3.52. TDH.SYS.CONFIG Leaf

Globally configure the Intel TDX module.

Table 6.205: TDH.SYS.CONFIG Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of an array of pointers, each containing the physical address of a single TDMR_INFO entry (see 4.2.4). 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 6.206: TDH.SYS.CONFIG Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.207: TDH.SYS.CONFIG Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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.	Addr. 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 6.208: 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	
TDX_SYS_SHUTDOWN	
TDX_SYSINIT_NOT_DONE	
TDX_SYSINITLP_NOT_DONE	
TDX_TDMR_ALREADY_INITIALIZED	
TDX_TDMR_OUTSIDE_CMRS	

6.3.53. UPDATED: 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 6.209: TDH.SYS.INFO Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 6.210: TDH.SYS.INFO Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.211: TDH.SYS.INFO Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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 actual number of CMRs.

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 actual number of CMRs.

Completion Status Codes

Table 6.212: 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	

6.3.54. TDH.SYS.INIT Leaf

Globally initialize the Intel TDX module.

Table 6.213: TDH.SYS.INIT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	Intel TDX module attributes		
	Bits	**Name**	**Description**
	63:0	RESERVED	Reserved: must be 0

Table 6.214: TDH.SYS.INIT Output Operands Definition

Operand	Description		
RAX	SEAMCALL instruction return code – see 6.3.1		
RCX	Extended error information part 1 In case of an 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 In case of an 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 In case of an 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
	63:32	MASK_EDX	Mask of the value returned by CPUID in EDX

Operand	Description		
R9	Extended error information part 4 In case of an 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 In case of an 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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.215: TDH.SYS.INIT Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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 6.216: 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	
TDX_INCORRECT_MSR_VALUE	
TDX_INVALID_WBINVD_SCOPE	
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	
TDX_SYS_SHUTDOWN	
TDX_SYSINIT_NOT_PENDING	

6.3.55. TDH.SYS.KEY.CONFIG Leaf

Configure the Intel TDX global private key on the current package.

Table 6.217: TDH.SYS.KEY.CONFIG Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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

Table 6.218: TDH.SYS.KEY.CONFIG Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.219: TDH.SYS.KEY.CONFIG Operands Information

Explicit/ Implicit	Reg.	Addr. 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, set PL.STATE to SYS_READY.

Completion Status Codes

Table 6.220: TDH.SYS.KEY.CONFIG Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_KEY_CONFIGURED	
TDX_KEY_GENERATION_FAILED	
TDX_OPERAND_BUSY	
TDX_SUCCESS	TDH.SYS.KEY.CONFIG is successful.
TDX_SYS_BUSY	
TDX_SYS_SHUTDOWN	
TDX_SYSINIT_NOT_DONE	

6.3.56. TDH.SYS.LP.INIT Leaf

Initialize the Intel TDX module at the current logical processor scope.

Table 6.221: TDH.SYS.LP.INIT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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

Table 6.222: TDH.SYS.LP.INIT Output Operands Definition

Operand	Description		
RAX	SEAMCALL instruction return code – see 6.3.1		
RCX	Extended error information part 1 In case of an inconsistent CPUID field 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).
RDX	Extended error information part 2 In case of an inconsistent CPUID field error, 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 In case of an inconsistent CPUID field error, 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
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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.223: TDH.SYS.LP.INIT Operands Information

Explicit/ Implicit	Reg.	Addr. 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. For example, the WBINVD scope must be the same as the scope the Intel TDX module was built to handle. 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 6.224: TDH.SYS.LP.INIT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_INCONSISTENT_CPUID_FIELD	

Completion Status Code	Description
TDX_INCONSISTENT_MSR	
TDX_INCORRECT_MSR_VALUE	
TDX_INVALID_PKG_ID	
TDX_SUCCESS	TDH.SYS.LP.INIT is successful.
TDX_SYS_BUSY	
TDX_SYS_SHUTDOWN	
TDX_SYSINIT_NOT_DONE	
TDX_SYSINITLP_DONE	

6.3.57. TDH.SYS.LP.SHUTDOWN Leaf

Initiate Intel TDX module shutdown, and prevent further SEAMCALLs on the current logical processor.

Table 6.225: TDH.SYS.LP.SHUTDOWN Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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

Table 6.226: TDH.SYS.LP.SHUTDOWN Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 marks the Intel TDX module as being shut down (if not already in this state) and prevents further SEAMCALLs on the current LP.

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 6.227: TDH.SYS.LP.SHUTDOWN Operands Information

Explicit/ Implicit	Reg.	Addr. 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. Mark the Intel TDX module as being shut down by setting PL.SYS_STATE to SYS_SHUTDOWN.
2. Prevent further SEAMCALLs on the current LP by setting the SEAM VMCS's HOST RIP field to the value of SYS_INFO_TABLE.SHUTDOWN_HOST_RIP (originally configured by the SEAMLDR).
3. Do a global EPT flush (INVEPT type 2).
   3.1. This is a defense-in-depth. In case of an Intel TDX module update, TDH.SYS.LP.INIT will do a global EPT flush.

Completion Status Codes

Table 6.228: TDH.SYS.LP.SHUTDOWN Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_SUCCESS	TDH.SYS.LP.SHUTDOWN is successful.

Completion Status Code	Description
TDX_SYS_BUSY	
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	

6.3.58. NEW: TDH.SYS.RD Leaf

Read a TDX Module global-scope metadata field.

Table 6.229: TDH.SYS.RD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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
RDX	Field identifier – see 4.8 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 6.230: TDH.MNG.RD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RDX	Next readable field identifier. A value of -1 indicates no next field identifier is available. In case of another error, as indicated by RAX, RDX returns -1.
R8	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.

TDH.SYS.RD reads a TDX Module global-scope metadata field.

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 0 as an input, until RDX returns 0.

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 6.231: TDH.SYS.RD Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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 6.2.1.1.
3. Return the next readable field identifier, or a value of 0 if none exists.
4. Return the field value.

Completion Status Codes

Table 6.232: TDH.SYS.RD 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	

6.3.59. NEW: TDH.SYS.RDALL Leaf

Read all host-readable TDX Module global-scope metadata fields.

Table 6.233: TDH.SYS.RDALL Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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
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. A value of 0 means start from the first field identifier.		

Table 6.234: TDH.SYS.RDALL Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 all applicable fields do not fit in the list, 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 = 0
2. STATUS = TDX_SUCCESS
3. While ((STATUS is not a non-recoverable error) && (NEXT_FIELD_ID != -1))
   - 3.1. Set LIST_BUFFER to the next 4K buffer
   - 3.2. Invoke TDH.SYS.RDALL(RDX = LIST_BUFFER, RDX = NEXT_FIELD_ID)
   - 3.3. STATUS = RAX, NEXT_FIELD_ID = R8

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 6.235: TDH.SYS.RDALL Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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 6.236: TDH.MNG.RDALL Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.MNG.RDALL is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_INITIALIZED	

6.3.60. TDH.SYS.TDMR.INIT Leaf

Partially initialize a Trust Domain Memory Region (TDMR) and its associated PAMT.

Table 6.237: TDH.SYS.TDMR.INIT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical base address of a TDMR (HKID bits must be 0)		

Table 6.238: TDH.SYS.TDMR.INIT Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.239: TDH.SYS.TDMR.INIT Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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. 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.
   4.1. PAMT_4K entries associated with a physical address that is within a reserved range are marked with PT_RSVD.
   4.2. Other PAMT_4K entries are marked with PT_NDA.
   4.3. PAMT_2M and PAMT_1G entries are marked with PT_NDA.
5. 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.
6. 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 6.240: TDH.SYS.TDMR.INIT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDH.SYS.TDMR.INIT is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TDMR_ALREADY_INITIALIZED	

6.3.61. UPDATED: TDH.VP.ADDCX Leaf

Add a TDCX page to memory as a child of a given TDVPR.

Table 6.241: TDH.VP.ADDCX Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 6.242: TDH.VP.ADDCX Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 TDCX page as a child of a given TDVPR.

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 6.243: TDH.VP.ADDCX Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDCX page	Blob	RW	Opaque	4KB	Exclusive	Shared	Shared
Explicit	RDX	HPA	TDVPR page	Blob	RW	Opaque	4KB	Exclusive	Shared	Shared
Implicit	N/A	HPA	TDR page	TDR	RW	Opaque	N/A	Shared	None	None
Implicit	N/A	N/A	TDCS structure	TDCS	R	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 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. Zero out the TDCX page contents using direct writes (MOVDIR64B).
10. Increment the VCPU's TDCX counter, and set a pointer in the parent TDVPR page to the new TDCX page.
11. Increment TDR.CHLDCNT.
12. Initialize the TDCX page metadata in PAMT.

Completion Status Codes

Table 6.244: TDH.VP.ADDCX Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.VP.ADDCX is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_TD_FATAL	
TDX_TD_FINALIZED	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_INITIALIZED	
TDX_TDCX_NUM_INCORRECT	
TDX_VCPU_STATE_INCORRECT	

6.3.62. UPDATED: TDH.VP.CREATE Leaf

Create a guest TD VCPU and its root TDVPR page.

Table 6.245: TDH.VP.CREATE Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	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 6.246: TDH.VP.CREATE Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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.

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 6.247: TDH.VP.CREATE Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions		
								Operand	Contain. 2MB	Contain. 1GB
Explicit	RCX	HPA	TDVPR page	Blob	RW	Opaque	4KB	Exclusive	Shared	Shared
Explicit	RDX	HPA	TDR page	TDR	RW	Opaque	4KB	Shared	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	R	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

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 have been initialized but not finalized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is INITIALIZED).
5. The TDVPR page metadata in PAMT must be correct (PT must be PT_NDA).

If successful, the function does the following:

6. Zero out the TDVPR page contents using direct write (MOVDIR64B).
7. Increment TDR.CHLDCNT.
8. Initialize the TDVPS management fields, which all reside in the TDVPR page.
9. Initialize the TDVPR page metadata in PAMT.

Completion Status Codes

Table 6.248: TDH.VP.CREATE Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
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_FINALIZED	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TD_NOT_INITIALIZED	

6.3.63. UPDATED: 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.

The following table details TDH.VP.ENTER input operands for **initial entry** or following a **previous asynchronous TD exit**.

Table 6.249: TDH.VP.ENTER Input Operands Definition for Initial Entry or Following a Previous Asynchronous TD Exit

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of the TD VCPU's TDVPR page (HKID bits must be 0)		

The following table details TDH.VP.ENTER input operands 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.

Table 6.250: TDH.VP.ENTER Input Operands Definition Following a Previous Trap-Like Asynchronous TD Exit with a Host Recoverability Hint

Operand	Description		
RAX	SEAMCALL instruction leaf and version numbers – see 6.3.1		
RCX	The physical address of the TD VCPU's TDVPR page (HKID bits must be 0)		
RDX	Host hints		
	Bit(s)	**Name**	**Description**
	0	HOST_RECOVERABILITY_HINT	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.
	Other	RESERVED	Must be 0

The following table details TDH.VP.ENTER input operands for following a **previous synchronous TD exit**.

Table 6.251: TDH.VP.ENTER Input Operands Definition Following a Previous TDCALL(TDG.VP.VMCALL)

Operand	Description
RAX	SEAMCALL instruction leaf and version numbers – see 6.3.1
RCX	The physical address of the TD VCPU's TDVPR page
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 6.4.15 for details. Else, the register value is not used as an input.

Operand	Description
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 6.4.15 for details. Else, the register value is not used as an input.

The following table details TDH.VP.ENTER output operands when an error occurs, and the interface function returns **without entering the TD**.

Table 6.252: TDH.VP.ENTER Output Operands Definition on Error (No TD Entry)

Operand	Description
RAX	SEAMCALL instruction return code
Other GPRs	Unmodified
Extended State	Any extended state that the TD is allowed to use (per TDCS.XFAM) may be cleared to its architectural INIT state.

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 6.253: TDH.VP.ENTER Output Operands Definition on Asynchronous TD Exits Following a TD Entry (with a VMX Architectural Exit Reason)

Operand	TD Exit Information	Description
RAX	Status and Exit Reason	SEAMCALL instruction return code The DETAILS_L1 field in bits 39:32 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 The DETAILS_L2 field in bits 31:0 contain the VMCS exit reason.
RCX	Exit Qualification	Format is similar to the VMCS exit qualification. When exit is due to EPT violation, bits 12-7 of the exit qualification are cleared to 0.
RDX	Extended Exit Qualification	Additional non-VMX, TDX-specific information – see 4.6.1
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.
RBX, RSI, RDI, R10 – R15	None	Cleared to 0
Extended State	Any extended state that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural INIT state.	

The following table details TDH.VP.ENTER output operands on when TD entry succeeds, and later an asynchronous TD exit occurs with a non-VMX TD exit status.

Table 6.254: TDH.VP.ENTER Output Operands Definition on Asynchronous TD Exits Following a TD Entry (with a non-VMX TD Exit Status)

Operand	TD Exit Information	Description
RAX	Status and Exit Reason	SEAMCALL instruction return code The DETAILS_L1 field in bits 39:32 may have the following values: • TDX_NON_RECOVERABLE_TD_CORRUPTED_MD
RCX, RDX, RBX, RBP, RDI, RSI, R8 – R15	Reserved	Cleared to 0
Extended State	Any extended state that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural INIT state.	

The following table details TDH.VP.ENTER output operands on 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 6.255: TDH.VP.ENTER Output Operands Definition on Asynchronous TD Exits Following a TD Entry (with Cross-TD Exit Details)

Operand	TD Exit Information	Description
RAX	Status and Exit Reason	SEAMCALL instruction return code The DETAILS_L1 field in bits 39:32 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.
RCX	Cross-TD Status	Status code of the error which caused the TD exit, using the same format as TDCALL instruction return code
RDX	Target TD	HPA of the TD which was the target of the cross-TD operation
RBX, RBP, RDI, RSI, R8 – R15	Reserved	Cleared to 0
Extended State	Any extended state that the TD is allowed to use (per TDCS.XFAM) is cleared to its architectural INIT state.	

The following table details TDH.VP.ENTER output operands on when TD entry succeeds, and later a **synchronous TD exit**, triggered by **TDG.VP.VMCALL**, occurs.

Table 6.256: TDH.VP.ENTER Output Operands Definition on TDCALL(TDG.VP.VMCALL) Following a TD Entry

Operand	Description
RAX	SEAMCALL instruction return code • The DETAILS_L2 field in bits 31:0 contain the VMCS exit reason, indicating TDCALL (77).

Operand	Description
RCX	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 6.4.15.
RBX, RDX, RBP, RDI, RSI, R8 – R15	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.
XMM0 – XMM15	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 INIT state.

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).
- Multiple MSRs are set as described below. In this table, Init(condition) means that the MSR is set to its INIT value if the condition is true, else the MSR is unmodified.

Table 6.257: MSRs that may be Modified by TDH.VP.ENTER

MSR Index Range (Hex)				
First (H)	Last (H)	Size (H)	MSR Architectural Name	MSR Preservation across TDH.VP.ENTER
0x00E1	0x00E1	0x1	IA32_UMWAIT_CONTROL	Init(virt. CPUID(7,0).ECX[5])
0x0186	0x018D	0x8	IA32_PERFEVTSELx	Init(PERFMON)
0x01A6	0x01A7	0x2	MSR_OFFCORE_RSPx	Init(PERFMON)
0x01C4	0x01C4	0x1	IA32_XFD	Init(virt. CPUID(0xD,0x1).EAX[4])
0x01C5	0x01C5	0x1	IA32_XFD_ERR	Init(virt. CPUID(0xD,0x1).EAX[4])
0x01D9	0x01D9	0x1	IA32_DEBUGCTL	INIT, except for the following bits which are preserved: Bit 1 (BTF) Bit 12 (FREEZE_PERFMON_ON_PMI) Bit 14 (FREEZE_WHILE_SMM)
0x0309	0x030C	0x4	IA32_FIXED_CTRx	Init(PERFMON)
0x0329	0x0329	0x1	IA32_PERF_METRICS	Init(PERFMON)
0x038D	0x038D	0x1	IA32_FIXED_CTR_CTRL	Init(PERFMON)
0x038E	0x038E	0x1	IA32_PERF_GLOBAL_STATUS	Init(PERFMON)
0x038F	0x038F	0x1	IA32_PERF_GLOBAL_CTRL	Init(PERFMON)
0x03F1	0x03F1	0x1	IA32_PEBS_ENABLE	Init(PERFMON)
0x03F2	0x03F2	0x1	MSR_PEBS_DATA_CFG	Init(PERFMON)
0x03F6	0x03F6	0x1	MSR_PEBS_LD_LAT	Init(PERFMON)
0x03F7	0x03F7	0x1	MSR_PEBS_FRONTEND	Init(PERFMON)
0x04C1	0x04C8	0x8	IA32_A_PMCx	Init(PERFMON)
0x0560	0x0560	0x1	IA32_RTIT_OUTPUT_BASE	Init(XFAM(8))
0x0561	0x0561	0x1	IA32_RTIT_OUTPUT_MASK_PTRS	Init(XFAM(8))
0x0570	0x0570	0x1	IA32_RTIT_CTL	Init(XFAM(8))
0x0571	0x0571	0x1	IA32_RTIT_STATUS	Init(XFAM(8))
0x0572	0x0572	0x1	IA32_RTIT_CR3_MATCH	Init(XFAM(8))
0x0580	0x0580	0x1	IA32_RTIT_ADDR0_A	Init(XFAM(8))
0x0581	0x0581	0x1	IA32_RTIT_ADDR0_B	Init(XFAM(8))
0x0582	0x0582	0x1	IA32_RTIT_ADDR1_A	Init(XFAM(8))

MSR Index Range (Hex)				
First (H)	Last (H)	Size (H)	MSR Architectural Name	MSR Preservation across TDH.VP.ENTER
0x0583	0x0583	0x1	IA32_RTIT_ADDR1_B	Init(XFAM(8))
0x0584	0x0584	0x1	IA32_RTIT_ADDR2_A	Init(XFAM(8))
0x0585	0x0585	0x1	IA32_RTIT_ADDR2_B	Init(XFAM(8))
0x0586	0x0586	0x1	IA32_RTIT_ADDR3_A	Init(XFAM(8))
0x0587	0x0587	0x1	IA32_RTIT_ADDR3_B	Init(XFAM(8))
0x0600	0x0600	0x1	IA32_DS_AREA	INIT
0x06A0	0x06A0	0x1	IA32_U_CET	Init(XFAM[11] | XFAM[12])
0x06A4	0x06A4	0x1	IA32_PL0_SSP	Init(XFAM[11] | XFAM[12])
0x06A5	0x06A5	0x1	IA32_PL1_SSP	Init(XFAM[11] | XFAM[12])
0x06A6	0x06A6	0x1	IA32_PL2_SSP	Init(XFAM[11] | XFAM[12])
0x06A7	0x06A7	0x1	IA32_PL3_SSP	Init(XFAM[11] | XFAM[12])
0x0985	0x0985	0x1	IA32_UINT_RR	Init(XFAM[14])
0x0986	0x0986	0x1	IA32_UINT_HANDLER	Init(XFAM[14])
0x0987	0x0987	0x1	IA32_UINT_STACKADJUST	Init(XFAM[14])
0x0988	0x0988	0x1	IA32_UINT_MISC	Init(XFAM[14])
0x0989	0x0989	0x1	IA32_UINT_PD	Init(XFAM[14])
0x098A	0x098A	0x1	IA32_UINT_TT	Init(XFAM[14])
0x0DA0	0x0DA0	0x1	IA32_XSS	Supervisor-mode feature bits of XFAM (bits 8, 16:10)
0x1200	0x12FF	0x100	IA32_LBR_INFO	Init(XFAM[15])
0x14CE	0x14CE	0x1	IA32_LBR_CTL	Init(XFAM[15])
0x14CF	0x14CF	0x1	IA32_LBR_DEPTH	Init(XFAM[15])
0x1500	0x15FF	0x100	IA32_LBR_FROM_IP	Init(XFAM[15])
0x1600	0x16FF	0x100	IA32_LBR_TO_IP	Init(XFAM[15])
0xC0000081	0xC0000081	0x1	IA32_STAR	INIT
0xC0000082	0xC0000082	0x1	IA32_LSTAR	INIT
0xC0000084	0xC0000084	0x1	IA32_FMASK	INIT
0xC0000102	0xC0000102	0x1	IA32_KERNEL_GS_BASE	INIT
0xC0000103	0xC0000103	0x1	IA32_TSC_AUX	INIT

Special Environment Requirements

The value read from IA32_TSC_ADJUST MSR must be the same as it was during TDH.SYS.INIT.

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)

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.

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.

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 Spec].

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 6.258: TDH.VP.ENTER Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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[7]	Shared[7]	Shared[7]
Implicit	N/A	HPA	TDR page	TDR	RW	Opaque	N/A	Shared[7]	N/A	N/A
Implicit	N/A	N/A	TDCS structure	TDCS	RW	Opaque	N/A	Shared(i)[7]	N/A	N/A
Implicit	N/A	N/A	TDCS.OP_STATE	OP_STATE	RW	Opaque	N/A	Shared[7]	N/A	N/A
Implicit	N/A	N/A	TDCS TLB Tracking Fields	N/A	RW	Opaque	N/A	Shared	N/A	N/A

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 must have been finalized (TDR.NUM_TDCX is the required number and TDCS.OP_STATE is RUNNING).

If successful, the function does the following:

5. Associate the VCPU with the current LP and update TD VMCS.
   - 5.1. Check that the VCPU has been initialized and is not being torn down.
   - 5.2. Atomically check that the VCPU is not associated with another LP, and associate it with the current LP.
   - 5.3. If the TD's ephemeral HKID has changed since last VM entry, update all TD VMCS physical pointers and the TD HKID execution control.
   - 5.4. Update the TD VMCS host state fields with any Intel TDX module LP-specific values.

If passed:

6. 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.
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. Set TDVPS.VCPU_STATE to VCPU_ACTIVE.
9. Restore guest TD state:
   - 9.1. If previous TD exit was due to a TDG.VP.VMCALL:
     - 9.1.1. 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.
     - 9.1.2. Set guest RAX to 0.
   - 9.2. Else (TD exit was an asynchronous exit):
     - 9.2.1. Restore CPU extended state from TDVPS (per TDCS.XFAM).
   - 9.3. Restore other guest state from TDVPS.

[7] The shared locking of TDVPS, TDR, TDCS, TDCS.OP_STATE (but not the TDCS epoch tracking fields) is for the whole duration of running in TDX non-root mode; the locks are released on TD exit.

10. Execute VMLAUNCH or VMRESUME depending on whether this VCPU has been launched on this LP since its last association with the LP (TVPS.VMLAUNCH).

Note: Logically, from the point of view of the host VMM, a successful TDH.VP.ENTER is terminated by the next TD exit.

Completion Status Codes

In case of successful execution (which resulted in the TD guest running and then exiting), the status code value in RAX is encoded the same as the VMX Exit reason – see the [TDX Module Spec] for details.

Table 6.259: TDH.VP.ENTER Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
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_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	Note the special case where the indicated operand is TLB_EPOCH. This may happen due to a conflict with TDH.MEM.TRACK. The host VMM may retry TDH.VP.ENTER.
TDX_OPERAND_INVALID	
TDX_OPERAND_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_NOT_FINALIZED	
TDX_TD_NOT_INITIALIZED	
TDX_VCPU_ASSOCIATED	
TDX_VCPU_STATE_INCORRECT	

6.3.64. TDH.VP.FLUSH Leaf

Flush the address translation caches and cached TD VMCS associated with a TD VCPU on the current logical processor.

Table 6.260: TDH.VP.FLUSH Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of a TDVPR page (HKID bits must be 0)		

Table 6.261: TDH.VP.FLUSH Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.262: TDH.VP.FLUSH Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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. Flush the TLB context and extended paging structure (EPxE) caches associated with the TD using INVEPT single-context invalidation (type 1).
5. Flush the cached TD VMCS content to TDVPS using VMCLEAR.
6. Mark the current VCPU as not associated with any LP.

7. Atomically decrement (using LOCK XADD) the associated VCPUs counter (TDCS.NUM_ASSOC_VCPUS).

Completion Status Codes

Table 6.263: TDH.VP.FLUSH Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_LIFECYCLE_STATE_INCORRECT	
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_PAGE_METADATA_INCORRECT	
TDX_SUCCESS	TDH.VP.FLUSH is successful.
TDX_SYS_NOT_READY	
TDX_SYS_SHUTDOWN	
TDX_VCPU_NOT_ASSOCIATED	

6.3.65. UPDATED: TDH.VP.INIT Leaf

Initialize the saved state of a TD VCPU.

Operands

Table 6.264: TDH.VP.INIT Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of a TDVPR page (HKID bits must be 0)		
RDX	Initial value of the guest TD VCPU RCX		

Table 6.265: TDH.VP.INIT Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 initialized the saved state of a VCPU in the TDVPR and TDPX pages.

VCPU Association: TDH.VP.INIT associates the target TD VCPU with the current LP – for details, see the [TDX Module Spec].

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 6.266: TDH.VP.INIT Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	4KB	Shared	N/A	N/A
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	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 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 a unique sequential identifier to the VCPU.
9. Initialize the VCPU state fields in the logical TDVPS structure (TDVPR and TDCX pages).
10. 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.

Completion Status Codes

Table 6.267: 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	
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_FINALIZED	
TDX_TD_KEYS_NOT_CONFIGURED	
TDX_TDCX_NUM_INCORRECT	
TDX_VCPU_ASSOCIATED	
TDX_VCPU_STATE_INCORRECT	
TDX_VCPU_STATE_INCORRECT	

6.3.66. UPDATED: TDH.VP.RD Leaf

Read a VCPU-scope metadata fields (control structure field) of a TD.

Table 6.268: TDH.VP.RD Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of a TDVPR page (HKID bits must be 0)		
RDX	Field identifier – see 4.8 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 6.269: TDH.VP.RD Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
RDX	For TDH.VP.RD version 0, RDX is unmodified. For TDH.VP.RD version 1 or higher, RDX returns the next readable field identifier. A value of -1 indicates no next field is available. In case of another error, RDX returns -1.
R8	Field content 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.RD reads a TDVPS field, given its field code. Reading is subject to the field's readability (per the TD's ATTRIBUTES.DEBUG bit).

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 Spec].

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 6.270: TDH.VP.RD Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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.
   - 7.1. Check that the VCPU has been initialized and is not being torn down.
   - 7.2. Atomically check that the VCPU is not associated with another LP, and associate it with the current LP.
   - 7.3. If the TD's ephemeral HKID has changed since last VM entry, update all TD VMCS physical pointers and the TD HKID execution control.
   - 7.4. Update the TD VMCS host state fields with any Intel TDX module LP-specific values.

If passed:

8. Read the control structure field using the algorithm described in 6.2.1.1.

Completion Status Codes

Table 6.271: TDH.VP.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_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_TD_NOT_INITIALIZED	
TDX_VCPU_ASSOCIATED	

Completion Status Code	Description
TDX_VCPU_STATE_INCORRECT	

6.3.67. UPDATED: TDH.VP.WR Leaf

Write a VCPU-scope metadata field (control structure field) of a TD.

Table 6.272: TDH.VP.WR Input Operands Definition

Operand	Description		
RAX	SEAMCALL instruction leaf number and version, see 6.3.1		
	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	The physical address of a TDVPR page (HKID bits must be 0)		
RDX	Field identifier – see 4.8 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 6.273: TDH.VP.WR Output Operands Definition

Operand	Description
RAX	SEAMCALL instruction return code – see 6.3.1
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 specific bits of the value (R8) are written as specified by the write mask (R9). Writing is subject to the field's writability (per the TD's ATTRIBUTES.DEBUG bit). Writing of specific fields is also subject to additional rules as detailed in 5.3.

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 Spec].

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 6.274: TDH.VP.WR Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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.
   - 7.1. Check that the VCPU has been initialized and is not being torn down.
   - 7.2. Atomically check that the VCPU is not associated with another LP, and associate it with the current LP.
   - 7.3. If the TD's ephemeral HKID has changed since last VM entry, update all TD VMCS physical pointers and the TD HKID execution control.
   - 7.4. Update the TD VMCS host state fields with any Intel TDX module LP-specific values.

If passed:

8. Write the control structure field and return its old value, using the algorithm described in 6.2.1.2.
   - 8.1. Writes of some fields are subject to rules, as detailed per field in 5.3 – 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 Spec]. If the write operation is illegal, 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 6.275: TDH.VP.WR Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_ADDR_RANGE_ERROR	

Completion Status Code	Description
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_OPERAND_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_NOT_INITIALIZED	
TDX_TD_VMCS_FIELD_NOT_INITIALIZED	
TDX_TDVPS_FIELD_NOT_WRITABLE	
TDX_VCPU_ASSOCIATED	
TDX_VCPU_STATE_INCORRECT	
TO BE COMPLETED	

6.4. UPDATED: 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.

6.4.1. 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 6.276: TDCALL Input Operands Definition

Operand	Description		
RAX	Leaf and version numbers, as defined in the [TDX Module Spec]. See Table 6.278 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 6.277: TDCALL Output Operands Definition

Operand	Description
RAX	Instruction return code, indicating the outcome of execution of the instruction – see the [TDX Module Spec] for details.
Other	See individual TDCALL leaf functions.

Table 6.278: 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

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.

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 6.279: TDCALL Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_SUCCESS	TDCALL is successful.
TDX_OPERAND_INVALID	Illegal leaf number
Other	See individual leaf functions

6.4.2. TDG.MEM.PAGE.ACCEPT Leaf

Accept a pending private page, and initialize the page to all-0 using the TD ephemeral private key.

Table 6.280: TDG.MEM.PAGE.ACCEPT Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 6.3.1		
	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
RCX	EPT mapping information:		
	Bits	**Name**	**Description**
	2:0	Level	Level of the Secure EPT entry that maps the private page to be accepted: either 0 (4KB) or 1 (2MB) – see 4.5.1.
	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 6.281: TDG.MEM.PAGE.ACCEPT Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 6.4.1
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.

Accept a pending private page, previously added by TDH.MEM.PAGE.AUG, into the TD. Initialize the page to 0.

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 6.282 TDG.MEM.PAGE.ACCEPT 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
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	RW	Opaque	N/A	Shared

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions
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[8], Transaction

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 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. Atomically (using LOCK CMPXCHG), check that the entry state is still PENDING, and set it to MAPPED.
    3.1. If failed (a concurrent host-side function may have changed the Secure EPT entry state), do a TD exit with an EPT Violation exit reason and a NOT_PENDING indication in the extended exit qualification.

Completion Status Codes

Table 6.283: TDG.MEM.PAGE.ACCEPT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_INVALID	
TDX_OPERAND_BUSY	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.

[8] Guest-side only

6.4.3. 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 6.284: TDG.MR.REPORT Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 6.3.1		
	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
RCX	1024B-aligned guest physical address of newly created report structure		
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 6.285: TDG.MR.REPORT Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 6.4.1
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.

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.

Additional REPORTDATA, a 64-byte value, is provided by the guest TD to be included in the TDG.MR.REPORT.

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 6.286: TDG.MR.REPORT Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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

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
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

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, the function does the following:

2. Assemble a report type structure based on the report sub type provided in R8.
3. Assemble the output report's TDINFO fields from the TDCS reported fields (ATTRIBUTES, XFAM, MRTD, MRCONFIGID, MROWNER, MROWNERCONFIG and RTMRs).
4. Calculate a SHA384 hash over TDINFO.
5. Execute SEAMREPORT to complete the output report, based on the input report data, the TDINFO hash calculated above and the report type structure.

If successful:

6. Write the output report to memory.

Completion Status Codes

Table 6.287: TDG.MR.REPORT Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.MR.REPORT is successful.

6.4.4. TDG.MR.RTMR.EXTEND Leaf

Extend a TDCS.RTMR measurement register.

Table 6.288: TDG.MR.RTMR.EXTEND Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 6.3.1		
	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
RCX	64B-aligned guest physical address of a 48B extension data buffer		
RDX	Index of the measurement register to be extended		

Table 6.289: TDG.MR.RTMR.EXTEND Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 6.4.1
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.

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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.290 TDG.MR.RTMR.EXTEND Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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
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

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 6.291: TDG.MR.RTMR.EXTEND Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.MR.RTMR.EXTEND is successful.

6.4.5. NEW: TDG.SERVTD.RD Leaf

As a service TD, read a metadata field (control structure field) of a target TD.

Table 6.292: TDG.SERVTD.RD Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 6.3.1		
	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
RCX	Binding handle		
RDX	Field identifier – see 4.8 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 195:128		
R13	Target TD's TD_UUID bits 255:196		

Table 6.293: TDG.SERVTD.RD Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 6.4.1
RDX	RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available. 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 In case of an error, as indicated by RAX, R10 is unmodified
R11	Updated target TD's TD_UUID bits 127:64 In case of an error, as indicated by RAX, R11 is unmodified
R12	Updated target TD's TD_UUID bits 195:128 In case of an error, as indicated by RAX, R12 is unmodified
R13	Updated target TD's TD_UUID bits 255:196 In case of an error, as indicated by RAX, R13 is unmodified

Operand	Description
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.

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 6.294 TDG.SERVTD.RD Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	None	Opaque	N/A	Shared(h)	Shared(h)	Shared(h)
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)	None	None
Implicit	N/A	N/A	Service TD's TDCS.RTMR	SHA384_ HASH	N/A	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	Target TD's TDCS structure	TDCS	R	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	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[9].
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.

[9] This value is a property of the TDX module and is the same for all TDs.

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 6.2.1.1**Error! Reference source not found.**.

Completion Status Codes

Note: Bit 60 (HOST_RECOVERABILITY_HINT) may be set by the host VMM if an error resulted in a trap-like TD exit followed by a TDH.VP.ENTER. See the [TDX Module Spec] for details.

Table 6.295: TDG.SERVTD.RD Completion Status Codes (Returned in RAX) Definition [TO BE COMPLETED]

Completion Status Code	Description
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.SERVTD.RD is successful.

6.4.6. NEW: TDG.SERVTD.WR Leaf

As a service TD, write a metadata field (control structure field) of a target TD.

Table 6.296: TDG.SERVTD.WR Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 6.3.1		
	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
RCX	Binding handle		
RDX	Field identifier – see 4.8 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 195:128		
R13	Target TD's TD_UUID bits 255:196		

Table 6.297: TDG.SERVTD.WR Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 6.4.1
R8	Previous contents of the field In case of an error, R8 returns 0.
R10	Updated target TD's TD_UUID bits 63:0 In case of an error, as indicated by RAX, R10 is unmodified
R11	Updated target TD's TD_UUID bits 127:64 In case of an error, as indicated by RAX, R11 is unmodified
R12	Updated target TD's TD_UUID bits 195:128 In case of an error, as indicated by RAX, R12 is unmodified
R13	Updated target TD's TD_UUID bits 255:196 In case of an error, as indicated by RAX, R13 is 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.

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 6.298 TDG.SERVTD.WR Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. 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	None	Opaque	N/A	Shared(h)	Shared(h)	Shared(h)
Implicit	N/A	N/A	TDR page	TDR	None	Opaque	N/A	Shared	Shared	Shared
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)	None	None
Implicit	N/A	N/A	Service TD's TDCS.RTMR	SHA384_ HASH	N/A	Opaque	N/A	Shared	N/A	N/A
Implicit	N/A	N/A	Target TD's TDCS structure	TDCS	R	Opaque	N/A	Shared(i)	None	None
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	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[10].
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.

[10] This value is a property of the TDX module and is the same for all TDs.

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 6.2.1.2.

Completion Status Codes

Note: Bit 60 (HOST_RECOVERABILITY_HINT) may be set by the host VMM if an error resulted in a trap-like TD exit followed by a TDH.VP.ENTER. See the [TDX Module Spec] for details.

Table 6.299: TDG.SERVTD.WR Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.VM.WR is successful.

6.4.7. NEW: TDG.SYS.RD Leaf

Read a TDX Module global-scope metadata field.

Table 6.300: TDG.SYS.RD Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 6.3.1		
	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
RDX	Field identifier – see 4.8 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 6.301: TDG.SYS.RD Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 6.4.1
RDX	Next readable field identifier. A value of -1 indicates no next field identifier is available. In case of another error, as indicated by RAX, RDX returns -1.
R8	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.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 0 as an input, until RDX returns 0.

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 6.302 TDG.SYS.RD Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions
There are no relevant memory operands.								

1. Read the requested field using the algorithm described in 6.2.1.1.
2. Return the next readable field identifier, or a value of 0 if none exists.
3. Return the field value.

Completion Status Codes

Table 6.303: TDG.SYS.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.SYS.RD is successful.

6.4.8. NEW: TDG.SYS.RDALL Leaf

Read all guest-readable TDX module global-scope metadata fields.

Table 6.304: TDG.SYS.RDALL Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 6.3.1		
	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
RDX	The GPA 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. A value of 0 means start from the first field identifier.		

Table 6.305: TDG.SYS.RDALL Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 6.4.1
R8	Next field identifier. A value of -1 means all applicable field identifiers have been returned in the metadata list. In case of another 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 all applicable fields do not fit in the list, 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 = 0
2. STATUS = TDX_SUCCESS
3. While ((STATUS is not a non-recoverable error) && (NEXT_FIELD_ID != -1))
   - 3.1. Set LIST_BUFFER to the next 4K buffer
   - 3.2. Invoke TDG.SYS.RDALL(RDX = LIST_BUFFER, RDX = NEXT_FIELD_ID)
   - 3.3. STATUS = RAX, NEXT_FIELD_ID = R8

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 6.306: TDG.SYS.RDALL Memory Operands Information Definition

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions
Explicit	RDX	GPA	Metadata List	MD_LIST	RW	Private	4096	None

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 6.307: TDG.SYS.RDALL Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.SYS.RDALL is successful.

6.4.9. UPDATED: TDG.VM.RD Leaf

Read a TD-scope metadata field (control structure field) of a TD.

Table 6.308: TDG.VM.RD Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 6.3.1		
	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
RCX	Reserved, must be 0		
RDX	Field identifier – see 4.8 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 6.309: TDG.VM.RD Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 6.4.1
RDX	For TDG.VM.RD version 0, RDX is unmodified. For TDG.VM.RD version 1 or higher, RDX returns the next readable field identifier. A value of -1 indicates no next field identifier is available. 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.
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.

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 6.310 TDG.VM.RD Memory Operands Information Definition

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

Explicit/ Implicit	Reg.	Addr. 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	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 6.2.1.1.

Completion Status Codes

5 **Table 6.311: TDG.VM.RD Completion Status Codes (Returned in RAX) Definition**

Completion Status Code	Description
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.VM.RD is successful.

6.4.10. UPDATED: TDG.VM.WR Leaf

Write a TD-scope metadata field (control structure field) of a TD.

Table 6.312: TDG.VM.WR Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 6.3.1		
	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
RCX	Reserved, must be 0		
RDX	Field identifier – see 4.8 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 6.313: TDG.VM.WR Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 6.4.1
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.

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 6.314 TDG.VM.WR Memory Operands Information Definition

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
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions
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 6.2.1.2.

Completion Status Codes

Table 6.315: TDG.VM.WR Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.VM.WR is successful.

6.4.11. UPDATED: 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 6.316: TDG.VP.CPUIDVE.SET Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 6.3.1		
	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
RCX	Controls whether CPUID executed by the guest TD will cause #VE unconditionally		
	Bits	**Name**	**Description**
	0	SUPERVISOR	Flags that when CPL is 0, a CPUID executed by the guest TD will cause a #VE unconditionally
	1	USER	Flags that when CPL > 0, a CPUID executed by the guest TD will cause a #VE unconditionally
	63:2	RESERVED	Reserved: must be 0

Table 6.317: TDG.VP.CPUIDVE.SET Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 6.4.1
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.

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.

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 6.318 TDG.VP.CPUIDVE.SET Memory Operands Information Definition

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
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

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 injection for CPUID for the current VCPU.

Completion Status Codes

Table 6.319: 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.

6.4.12. UPDATED: TDG.VP.INFO Leaf

Get guest TD execution environment information.

Table 6.320: TDG.VP.INFO Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 6.3.1		
	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

Table 6.321: TDG.VP.INFO Output Operands Definition

Operand	Description		
RAX	TDCALL instruction return code – see 6.4.1 – 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 Spec] chapter discussing general aspects of the Intel TDX Module API.

Table 6.322: TDG.VP.INFO Memory Operands Information Definition

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
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

Completion Status Codes

Table 6.323: TDG.VP.INFO Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_SUCCESS	TDG.VP.INFO is successful.

6.4.13. NEW: TDG.VP.RD Leaf

Read a VCPU-scope metadata field (control structure field) of a TD.

Table 6.324: TDG.VP.RD Input Operands

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 6.3.1		
	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
RCX	Reserved, must be 0		
RDX	Field identifier – see 4.8 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 6.325: TDG.VP.RD Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 6.4.1
RDX	Next readable field identifier. A value of -1 indicates no next field identifier is available. In case of another error, as indicated by RAX, RDX returns -1.
R8	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.VP.RD reads a VCPU-scope metadata field (control structure field) of a 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 6.326 TDG.VP.RD Memory Operands Information Definition

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
Implicit	N/A	N/A	TDCS structure	TDCS	R	Opaque	N/A	Shared(i)

Explicit/ Implicit	Reg.	Addr. Type	Resource	Resource Type	Access	Access Semantics	Align. Check	Concurrency Restrictions
Implicit	N/A	N/A	TDVPS structure	TDVPS	RW	Opaque	N/A	Shared

If the memory operand checks per the table above pass:

1. Read the control structure field using the algorithm described in 6.2.1.1.

Completion Status Codes

Table 6.327: TDG.VP.RD Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.VM.RD is successful.

6.4.14. 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 6.328: TDG.VP.VEINFO.GET Input Operands Definition

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 6.3.1		
	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

Table 6.329: TDG.VP.VEINFO.GET Output Operands Definition

Operand	Description		
RAX	TDCALL instruction return code – see 6.4.1		
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
	63:32	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**
	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
	In case of an error, R10 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.

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 6.330: TDG.VP.VEINFO.GET Memory Operands Information Definition

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
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

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 6.331: 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_SUCCESS	TDG.VP.VEINFO.GET is successful.

6.4.15. TDG.VP.VMCALL Leaf

Perform a TD Exit to the host VMM.

Table 6.332: TDG.VP.VMCALL Input Operands Definition

<table>
<tr><th>Operand</th><th colspan="3">Description</th></tr>
<tr><td rowspan="4">RAX</td><td colspan="3">TDCALL instruction leaf number and version, see 6.3.1</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 TDCALL interface function</td></tr>
<tr><td>23:16</td><td>Version Number</td><td>Selects the TDCALL interface function version</td></tr>
<tr><td>63:24</td><td>Reserved</td><td>Must be 0</td></tr>
<tr><td rowspan="5">RCX</td><td colspan="3">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.</td></tr>
<tr><td>Bits</td><td>Name</td><td>Description</td></tr>
<tr><td>15:0</td><td>GPR Mask</td><td>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
Bit 6: RSI
Bit 7: RDI
Bits 15:8: R15 – R8</td></tr>
<tr><td>31:16</td><td>XMM Mask</td><td>Controls the transfer of XMM15 – XMM0 register values</td></tr>
<tr><td>63:32</td><td>Reserved</td><td>Reserved: must be 0</td></tr>
<tr><td>RBX, RDX, RBP, RSI, RDI, R8 – R15</td><td colspan="3">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.</td></tr>
<tr><td>XMM0 – XMM15</td><td colspan="3">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.</td></tr>
</table>

Table 6.333: TDG.VP.VMCALL Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code: returns a constant value of TDX_SUCCESS (0)

Operand	Description
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.
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.

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 6.334: TDG.VP.VMCALL Memory Operands Information Definition

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
Implicit	N/A	N/A	TDCS structure	TDCS	None	Opaque	N/A	Shared(i)
Implicit	N/A	N/A	TDVPS structure	TDVPS	R/W	Opaque	N/A	Shared

1. Save guest TD CPU state to TDVPS (including TD VMCS):
   1.1. 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.
   1.2. 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.
   1.3. Advance the saved RIP to the instruction following TDCALL.
2. Adjust the TDCS TLB tracking counters.
3. Release the shared locking – acquired on TDH.VP.ENTER of TDR, TDCS and TDVPS.
4. Load host VMM state:
   4.1. Clear the extended state except XMM (per TDCS.XFAM) to synthetic INIT values.
   4.2. As controlled by RCX, either clear or set to the guest TD's value the state of XMM0 – XMM15.
   4.3. 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.
   4.4. Set RCX to the guest TD's value.
   4.5. Set RAX to the TDCALL exit reason.
   4.6. Restore other host VMM state – saved during TDH.VP.ENTER.
5. 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 6.335: 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.

6.4.16. NEW: TDG.VP.WR Leaf

Write a VCPU-scope metadata field (control structure field) of a TD.

Table 6.336: TDG.VP.WR Input Operands

Operand	Description		
RAX	TDCALL instruction leaf number and version, see 6.3.1		
	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
RCX	Reserved, must be 0		
RDX	Field identifier – see 4.8 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 6.337: TDG.VP.WR Output Operands Definition

Operand	Description
RAX	TDCALL instruction return code – see 6.4.1
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.VP.WR writes a VCPU-scope metadata field (control structure field) of a 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 6.338 TDG.VM.WR Memory Operands Information Definition

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
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

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 6.2.1.2.

Completion Status Codes

Table 6.339: TDG.VP.WR Completion Status Codes (Returned in RAX) Definition

Completion Status Code	Description
TDX_OPERAND_BUSY	
TDX_OPERAND_INVALID	
TDX_SUCCESS	TDG.VP.WR is successful.