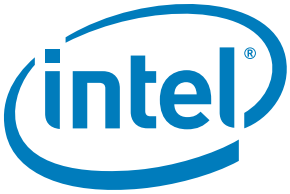

Intel® Trust Domain Extensions - SEAM Loader (SEAMLDR) Interface Specification

343755-001US
SEPTEMBER 2020

Revision History

Revision	Description	Date
343755-001	Initial release of document.	September 2020

TABLES

FIGURES

PAGE

CHAPTER 1
SECURE ARBITRATION MODE (SEAM)

1.1 OVERVIEW

Secure Arbitration Mode (SEAM) is an extension to the Virtual Machines Extension (VMX) architecture to define a new, VMX root operation called SEAM VMX root and a new VMX non-root operation called SEAM VMX non-root. Collectively, the SEAM VMX root and SEAM VMX non-root execution are called operation in SEAM.

SEAM VMX root operation is designed to host a CPU-attested, software module called the Intel® Trust-Domain-Extensions (Intel® TDX) module to manage virtual machine (VM) guests called **Trust Domains (TD)**. Virtual machines launched/resumed from SEAM VMX root operation are TDs, and VMs launched/resumed from legacy VMX root operation are legacy VMs. Intel TDX modules use the SEAM instruction set extensions to help protect the confidentiality and integrity of TD memory contents and CPU state from all other software, including the hosting Virtual Machine Monitor (VMM), unless explicitly shared by the TD itself.

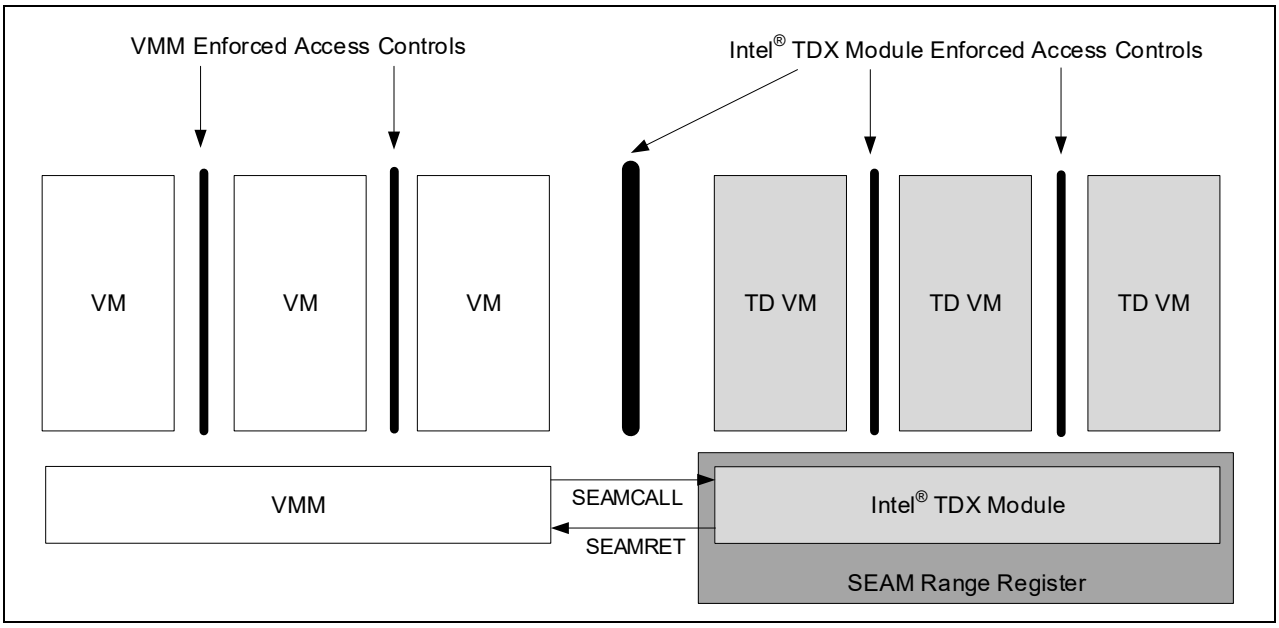

Figure 1-1. Intel® Trust Domain Extension Components

The Intel TDX module helps implements the functions to build, tear down, and start execution of TD VMs. The VMM is designed to provide the memory resources to build the TD and schedule the TD executions using the API provided by the Intel TDX module. The Intel TDX module is formatted to execute out of a range of memory defined using a SEAM range register (SEAMRR) interface. The IA32_SEAMRR_PHYS_BASE MSR helps define the 32-MB aligned base address for the SEAM memory range. The IA32_SEAMRR_PHYS_MASK MSR contains a mask that can determine the address range protected by the SEAMRR interface.

Intel provides a Secure Arbitration Mode Loader (SEAMLDR) ACM to help initialize the SEAM range, set up the SEAM transfer VMCS structure, and load the Intel TDX module into this protected range of memory. The OS can launch the SEAMLDR ACM using the GETSEC[ENTERACCS] instruction if the SEAMRR range enable bit (bit 11) of IA32_SEAMRR_PHYS_MASK MSR is 1. The SEAMLDR aims to measure and verify the Intel TDX module against its signature structure and record its security version number (SVN), measurements, and identity into CPU registers that are accessible only to the SEAMLDR ACM.

The SVN of the SEAMLDR ACM itself is reported in the IA32_SGX_SVN_STATUS MSR in bits 63:56. OS/VMMs that launch an ACM such as SINIT or SEAMLDR are expected to read the IA32_SGX_SVN_STATUS MSR to determine whether the ACM can be launched or if a new ACM is needed. If either the Intel® Software-Guard-Extensions (Intel® SGX) SVN of the ACM value in the ACM's header is greater than the value reported by the

IA32_SGX_SVN_STATUS MSR or the lock bit in the IA32_SGX_SVN_STATUS MSR is not set, then the OS/VMM can safely launch the ACM. If the Intel SGX SVN value reported in the corresponding component of the IA32_SGX_SVN_STATUS MSR is greater than the Intel SGX SVN value in the ACM's header, and if bit 0 of the IA32_SGX_SVN_STATUS MSR is 1, then the OS/VMM would not launch that version of the ACM. It would obtain an updated version of the ACM either from the BIOS or from an external resource.

The SEAMLDR ACM is designed to follow the steps below to load or update the Intel TDX module into SEAMRR:

1. Verify startup state; basic checks on current state of platform and parameters passed by OS/VMM.
2. Signature structure verification; verify signature on signature structure.
3. Load module image; load Intel TDX module image into SEAMRR, measure the image and verify the measurements match the signature structure.
4. Set up data and stack regions for the Intel TDX module.
5. Set up SEAM transfer VMCS array used by SEAMCALL and SEAMRET instructions.
6. Record SEAM identity into CPU measurement registers and update its load status.
7. Exit to OS/VMM.

CHAPTER 2
SEAMLDR DATA STRUCTURES

2.1 SEAM_SIGSTRUCT: INTEL® TDX MODULE SIGNATURE STRUCTURE

All fields in the SEAM_SIGSTRUCT (e.g., MODULUS, SIGNATURE, etc.) are in little-endian form.

Table 2-1. SEAM_SIGSTRUCT: Intel® TDX Module Signature Structure

Field Name	Offset	Size	Description	Signed?
SIGNATURE STRUCTURE HEADER				
HEADER_TYPE	0	4	Module type; must be 6. Generic FW.	Y
HEADER_LENGTH	4	4	Header Length; must be 0xE1 (900 bytes). The length includes the following: ▪ Header: 128 B ▪ MODULUS: 384 B ▪ EXPONENT: 4 B ▪ SIGNATURE: 384 B The bytes following this Header (offset 900 onwards) are the body of the signed module.	Y
HEADER_VERSION	8	4	Structure Version Bits 31:16 - Major Version Bits 15:0 - Minor Version Must be 0x00010000.	Y
MODULE_TYPE	12	4	Module Type Bits 30:0 must be 0. Bit 31: Debug/Production Signed ▪ 0: Production ▪ 1: Debug	Y
MODULE_VENDOR	16	4	Module Vendor Intel: 8086h, Non-Intel: 0000h	Y
DATE	20	4	Build date: yyyymmdd format (yyyy: four-digit year, mm: 1 to 12, dd: 1 to 31)	Y
SIZE	24	4	Size of entire module (i.e., SEAM_SIGSTRUCT) in DWORDS; must be 0x200.	Y
KEY_SIZE	28	4	RSA public key size; must be 0x60.	Y
MODULUS_SIZE	32	4	RSA public key modulus size; must be 0x60.	Y
EXPONENT_SIZE	36	4	RSA public key exponent size; must be 1.	Y
RESERVED	40	88	Reserved; must be 0.	Y
MODULUS AND SIGNATURE				
MODULUS	128	384	Module public key (3072 bits) represented as a byte string of length 384, with the most significant byte at offset 511, i.e., in little-endian format.	N
EXPONENT	512	4	RSA exponent. Must be 2^16+1.	N

Table 2-1. SEAM_SIGSTRUCT: Intel® TDX Module Signature Structure

Field Name	Offset	Size	Description	Signed?
SIGNATURE	516	384	RSA signature represented as a byte string of length 384, with the most significant byte at offset 899, i.e., in little-endian format. Signature hash covers (Header (128B) || Body) in that order.	N
Intel TDX MODULE CONFIGURATION PARAMETERS (Body)				
SEAMHASH	900	48	Intel TDX module SHA384 hash.	Y
SEAMSVN	948	2	Intel TDX module SVN.	Y
ATTRIBUTES	950	8	TDX module attributes (non-Intel).	Y
RIP_OFFSET	958	4	Offset of Intel TDX module entry point.	Y
NUM_STACK_PAGES	962	1	Stack size per thread in units of (# of 4K pages) - 1.	Y
NUM_TLS_PAGES	963	1	TLS size per thread in units of (# of 4K pages) - 1.	Y
NUM_KEYHOLE_PGS	964	2	Keyhole pages in units of (# of 4K pages) - 1.	
MIN_GLB_DATA_PAGES	966	2	Minimum number of global data pages needed by the module in units of (# of 4K pages) - 1.	
RESERVED	968	56	Reserved; must be 0.	Y
CPUID_TABLE_SIZE	1024	4	Number of entries in CPUID table. Valid values are 0 through 255. Setting to 0 matches all CPUs.	Y
CPUID_TABLE	1028	1020	Table of supported CPU version numbers as returned by CPUID.1.EAX. Stepping number is ignored for the match.	Y

2.2 SEAMLDR_PARAMS: SEAMLDR ACM PARAMETERS

The SEAMLDR_PARAMS structure is used by the OS/VMM loader to help provide the pointer to the Intel TDX module signature structure and the list of Intel TDX module image pages to be loaded or updated. The version 0 SEAMLDR_PARAMS structure is 4K bytes in size and formatted as described in Table 2-2, and, as designed, the SEAMLDR_PARAMS structure must be aligned to 4K bytes.

Table 2-2. SEAM_SIGSTRUCT: Intel® TDX Module Signature Structure

Field Name	Offset (Bytes)	Size (Bytes)	Description
VERSION	0	4	Structure version; must be 0.
SCENARIO	4	4	Scenario for which SEAMLDR was invoked: ▪ 0: LOAD. Load Intel TDX module. ▪ 1: UPDATE. Update previously loaded module.
SIGSTRUCT_PA	8	8	4-KByte aligned, physical address of the Intel TDX module signature structure.
RESERVED	16	104	Reserved; must be 0.
NUM_MODULE_PAGES	120	8	Intel TDX module size in number of 4-KByte pages. Valid range 1 to 496 for version 0.
MOD_PAGES_PA_LIST	128	8 * 496	Array of 4-KByte aligned, physical address pointers to Intel TDX module pages.

2.3 ERROR CODES

The SEAMLDR returns error codes in the format: 0x8000 0000 cccc eeee. Here, cccc is the field that describes the error class, and the field eeee specifies the error code. The error classes are described in Table 2-3.

Table 2-3. Error Classes

Error Class	Error Class Name	Description
0000	ECPARAM	Parameter validation errors. These errors are usually indicative of errors in the software that invokes the SEAMLDR.
0001	ECPLAT	Platform configuration errors. These are usually indicative of miscon-figuration of the platform. This might be due to BIOS errors or unsupported hardware configurations.
0002	ECIMG	Module image verification errors. These are usually indicative of corruptions in the module image leading to errors like signature verifi-cation error, etc.

The list of errors codes returned by the SEAMLDR are described in Table 2-4.

Table 2-4. Error Codes

Error Code	Error Name	Description
0x8000 0000 0000 0000	EBADPARAM	Bad input parameter.
0x8000 0000 0000 0001	EMODBUSY	Intel TDX module has not been shut down.
0x8000 0000 0000 0002	ELDINPROG	Software attempted to invoke SEAMLDR simultaneously on multiple logical processors.
0x8000 0000 0001 0000	EBADRANGE	SEAMRR range is not valid.
0x8000 0000 0001 0001	EBADPLATF	Unsupported platform configuration.
0x8000 0000 0001 0002	ENOMEM	The module does not fit within the SEAM range constraints.
0x8000 0000 0001 0003	EUNSPECERR	Unspecified platform configuration error.
0x8000 0000 0002 0000	EBADSIG	Malformed module signature structure, or signature verification failed.
0x8000 0000 0002 0001	EBADHASH	Module image hash verification failed.
0x8000 0000 0002 0002	EUNSUPCPU	Module does not support one or more CPUs in the platform.

CHAPTER 3
INTEL® TDX MODULE LOAD AND UPDATE

The OS/VMM loader is designed to perform the following actions to load or update a previously loaded Intel® TDX module:

1. Find the Intel TDX module image and SEAM signature structure and copy the image and signature structure to memory. The memory for the Intel TDX module image and SEAM signature structure need not be contiguous and can be above 4 GB.
2. Find the SEAMLDR ACM image file and copy it to memory. The memory for the SEAMLDR image must be contiguous, physical memory and must be below 4 GB.
3. Verify if module supports all the processors in the platform. If the SEAM_SIGSTRUCT. CPUID_TABLE_SIZE is 0, then the module supports all processors that support SEAM mode, and the next step can be skipped.
4. Collect CPUID.1.EAX from all sockets in the platform and mask off the bits 3:0, i.e., the stepping number. Compare each of these values to each valid entry in SEAM_SIGSTRUCT. CPUID_TABLE. The number of valid entries in SEAM_SIGSTRUCT. CPUID_TABLE is from 0 to (SEAM_SIGSTRUCT. CPUID_TABLE_SIZE - 1). If the module does not support all the processors in the platform, then this module cannot be loaded. The OS/VMM might need to find a module supported by the platform from the module vendor.
5. Ensure that the SEAMLDR ACM is compatible with the platform using the same rules as other ACMs. However, SEAMLDR does not have dependence on Intel® TXT-supported chipset and TPM. So, filtering by ChipsetID or TPM information tables is not required (the SEAMLDR will have empty ChipsetID and TPM information tables). The steps to be performed are as follows:
   - Step 1: Verify it is SEAMLDR ACM by evaluating module type.
   - Step 2: Verify platform type match.
   - Step 3: Verify AC module and chipset production flags match.
   - Step 4: Match CPUID.FMS and PLATFORM_ID.
6. The SEAMLDR ACM must be launched on a logical processor marked BSP (i.e., where IA32_APIC_BASE.BSP is 1). All other logical processors in the socket on which SEAMLDR will be launched must be in the WAIT-FOR-SIPI state. If they are not already in the WAIT-FOR-SIPI state (for example, coming out of reset), then the OS might need to generate INIT IPI to all other logical processors to place them in the WAIT-FOR-SIPI state.
7. When invoking the UPDATE scenario, the SEAMLDR ACM also requires that the OS/VMM loader has invoked the shutdown function provided by the Intel TDX module, and VMXOFF has been executed on all logical processors in the platform.
8. Allocate a page of memory to use as SEAMLDR_PARAMS and initialize the structure as described in Table 3-1.

Table 3-1. Settings to Initialize Intel® TDX Module Signature Structure

Field Name	Offset (Bytes)	Size (Bytes)	Description
VERSION	0	4	Set to 0.
SCENARIO	4	4	Set to 0 to LOAD an Intel TDX module for first time. Otherwise, set to 1 to UPDATE a previously loaded Intel TDX module.
SIGSTRUCT_PA	8	8	Program to the 4-KByte aligned, physical address of page that has the SEAM_SIGSTRUCT of the module.
RESERVED	16	104	Set to 0.
NUM_MODULE_PAGES	120	8	Set to number of pages of Intel TDX module image.
MOD_PAGES_PA_LIST	128	8 * 496	Program this array with 4-KByte aligned, pointer to the pages of the module image. The array must be filled such that the first entry points to the first page of the module, second entry to the second page, and so on in that order.

9. Prepare to launch SEAMLDR ACM and set up parameters for the SEAMLDR in registers as follows:
   a. R8 = 4-KByte aligned, SEAMLDR_PARAMS structure physical address.
   b. R9 = GDT base to be set up by the SEAMLDR when returning to the OS.
   c. R10 = RIP where control is transferred to OS following SEAMLDR execution.
   d. R11 = CR3 value that should be established when returning control to OS.
   e. EBX = SEAMLDR ACM physical address base.
   f. ECX = SEAMLDR ACM size.
   g. EAX = 2; leaf number for the ENTERACCS function of the GETSEC instruction.
10. OS/VMM loader should launch SEAMLDR ACM in IA32e mode with 4-level or 5-level page tables. OS/VMM loader invokes GETSEC[ENTERACCS] to launch the SEAMLDR ACM.
11. SEAMLDR will set up CR3 on exit with the OS/VMM page table as specified at launch by VMM through register R11. Certain processor states are not restored on EXITAC, and the OS/VMM loader must re-establish such states when control returns from the SEAMLDR.
12. SEAMLDR ACM returns status in register R9.
13. TXT MLE that launches SEAMLDR ACM using ENTERACCS should be aware of following:
   a. SEAMLDR ACM does not close the TPM locality 2 or the TXT private space if the platform is post-SENTER as indicated by LT.STS [0] == 0x01. An MLE that launches SEAMLDR should be aware of this and re-evaluate the TPM locality 2 and TXT private space status following execution of the SEAMLDR.
   b. ENTERACCS unconditionally unmasks SMI, INIT, and NMI. Such an MLE must only launch SEAMLDR ACM after the MLE has enabled SMI (or there is no STM/PPAM, and so SMI are enabled by SINIT).

3.1 CPU STATE FOLLOWING SEAMLDR ACM EXECUTION

Table 3-2 describes the intended, CPU state after SEAMLDR ACM execution.

Table 3-2. CPU State After SEAMLDR ACM Execution

Register	CPU State After EXITAC from SEAMLDR
CR0	PE=1, PG=1, NE=1, CD=0, NW=0; other bits cleared.
CR3	Provided by OS/VMM in R11.
CR4	PAE=1, SMXE=1, MCE=0, PGE and LA57 are unchanged; all other bits are cleared.
RFLAGS	0x0002
IA32_EFER	LME=1, LMA=1, NXE=1; other bits cleared.
IA32_PAT	Reset value.
RIP	Provided by OS/VMM in R10.
RAX	0x3 (EXITAC)
RBX	Value provided by OS/VMM in R10.
RCX, RDX, RBP, R10, R11	0
R8	Value provided by OS/VMM in R11.
R9	Load status returned by SEAMLDR: success/error code.
Other GPRs, including RSP	0
CS	Flat
DS	Flat
GDTR	Base = Provided by OS/VMM in R9; Limit and Selector are unchanged.
IDTR	Unchanged

Table 3-2. CPU State After SEAMLDR ACM Execution (Continued)

Register	CPU State After EXITAC from SEAMLDR
DR7	0x400
IA32_DEBUGCTL	0x0
IA32_PERF_GLOBAL_CTRL	0x0
IA32_PEBS_ENABLE	0x0
IA32_RTIT_CTRL	TraceEn = 0, other bits are unchanged.
IA32_LBR_CTRL	0x0
IA32_MISC_ENABLES	0x8 (only thermal throttling is enabled)
Other registers (including MSRs, XCR0, XSAVES-able registers, DRs)	Unchanged

3.2 HANDLING LOAD SCENARIO ERRORS

If the class of error is ECPARAM or ECPLAT, these are usually indicative of BIOS or OS/VMM software errors. Retrying the load will likely not succeed, and thus these errors can be fatal to the load process.

If the class of error is ECIMG, the OS/VMM might attempt to download or install the module image and associated signature structure again. If there was an inadvertent corruption that occurred on the previously downloaded image, then retrying with a newly downloaded image might lead to success.

3.3 HANDLING UPDATE SCENARIO ERRORS

If the class of error is ECPARAM, these are usually indicative of OS/VMM software errors. Retrying the update will likely not succeed, and thus these errors are fatal to the update process.

If the class of error is ECIMG, then the module image is either corrupted or was signed incorrectly. Retrying the update with this module image will likely not succeed. The OS/VMM might attempt to verify the signature and the module image hash. If the signature or module image hash are in error, this indicates a corruption of the module, and the OS/VMM might attempt to download the image again and retry with the newly downloaded image. If the image was not corrupted, then, by design, it has been signed incorrectly or the image does not support the platform on which it is being loaded. In this case retrying with the same image will likely not succeed, and thus this error can be fatal to the update process.

If the class of error is ECPLAT, the module is not compatible with the current, platform configuration. Retrying the update with this module image will likely not succeed, and thus this error can be fatal to the update process.

For both ECIMG and ECPLAT error classes, the OS/VMM might choose to use the module that was previously loaded. Since the previous version of the module was successfully loaded, attempting to do an update with the previous module image might be successful.