14.Debug and Profiling Architecture

The Intel TDX module debug architecture includes the following debug facilities:

On-TD Debug: Facilities for debugging a guest TD using software that runs inside the TD

Off-TD Debug: Facilities for debugging a guest TD, configured in debug mode, using software that runs outside the TD

14.1. On-TD Debug

Intel SDM, Vol. 3, 17 Debug, Branch Profile, TSC and Intel Resource Director Technology (Intel RDT) Features

14.1.1. Overview

On-TD debug is the normal mode used to debug guest TD software. A debug agent resides inside the guest TD, and it can interact with external entities (e.g., a debugger) via standard I/O interfaces. The Intel TDX module is designed to virtualize and isolate TD debug capabilities from the host VMM and other guest TDs or legacy VMs. On-TD debug can be used for production or debug TDs – i.e., regardless of the guest TD's ATTRIBUTES.DEBUG state.

Guest TDs are allowed to use almost all architectural debug features supported by the processor, e.g.:

- Single stepping
- Code, data and I/O breakpoints
- INT3
- Bus lock detection
- DR access detection
- TSX debug

However, the TDX architecture does not allow guest TDs to toggle IA32_DEBUGCTL uncore PMI enabling bit (13).

Guest TDs are allowed to use almost all architectural tracing features, e.g.:

- LBR (if allowed by the TD's XFAM, see 10.5)
- PT (if allowed by the TD's XFAM, see 10.5)
- BTS

However, the TDX architecture does not allow guest TDs to use BTM.

14.1.2. Generic Debug Handling

14.1.2.1. *Context Switch*

By design, the Intel TDX module context-switches all debug/tracing state that the guest TD is allowed to use.

- DR0-3, DR6 and IA32_DS_AREA MSR are context-switched in TDH.VP.ENTER and TD exit flows.
- RFLAGS, IA32_DEBUGCTL MSR and DR7 are saved and cleared on VM exits from the guest TD and restored on VM entry to the guest TD.
- Pending debug traps are natively saved on VM exits from the guest TD and reloaded on VM entries using the TD VMCS PDE field.

14.1.2.2. *IA32_DEBUGCTL MSR Virtualization*

Intel SDM, Vol. 3, 17.4.1 IA32_DEBUGCTL MSR

By design, IA32_DEBUGCTL access by the guest TD is restricted as follows:

- Guest TD attempts to set any of the architecturally-reserved bits 63:15 and 5:2 result in a #GP(0).
- Guest TD attempts to set TDX-disallowed values result in a #VE. This includes the following cases:
  - Enable Uncore PMI by setting bit 13 to 1 (see 14.4 below).
  - Enable BTM by setting bits 7:6 to 0x1 (see details in 14.1.3 below).
- Uncore PMI is virtualized as disabled; bit 13 is read as 0 (see 14.4 below).

14.1.3. Debug Feature-Specific Handling

The following table discusses how specific debug features are handled.

Table 14.1: Debug Feature-Specific Handling

Debug Feature	How the Feature is Controlled	Handling
Hardware Breakpoints	• DR7, DR0-3 and DR6	No special handling: DRs are context-switched.
General Detect	• DR7 bit 13 (GD)	No special handling: DR7 is context-switched.
TSX Debug	• DR7 bit 11 (RTM) • IA32_DEBUGCTL bit 15 (RTM)	No special handling: DR7 and IA32_DEBUGCTL are context-switched.
Single Stepping	• RFLAGS bits 18 (Trap Flag) and 16 (Resume Flag) • IA32_DEBUGCTL bit 1 (BTF)	No special handling: RFLAGS and IA32_DEBUGCTL are context-switched.
Bus-Lock Detection	• IA32_DEBUGCTL bit 2 (BUS_LOCK_DETECT)	No special handling: IA32_DEBUGCTL is context-switched.
Software Breakpoints (INT1, INT3)	None	No special handling: software breakpoints are stateless.
Branch Trace Message (BTM)	• IA32_DEBUGCTL bits 6 (TR) and 7 (BTS)	Not allowed: when a guest TD attempts to set IA32_DEBUGCTL[7:6] to 0x1, the Intel TDX module injects a #VE (see 14.1.2 above). In debug mode (ATTRIBUTES.DEBUG == 1), the host VMM is allowed to activate BTM by setting the above bits to 0x1.
Branch Trace Store (BTS)	• IA32_DEBUGCTL bits 6 (TR), 7 (BTS), 8 (BTINT), 9 (BTS_OFF_OS) and 10 (BTS_OFF_USR)	No special handling: IA32_DEBUGCTL and IA32_DS_AREA are context-switched. **Notes:** • The guest TD can configure BTS to raise PMI on buffer overflow (by setting BTINT = 1). However, since PMIs are virtualized by the host VMM, the guest TD should be ready to handle spurious, delayed and dropped PMIs. See Perfmon discussion in 14.2 below. • BTS may allow the guest TD to hang the machine if BTS record generation causes a #PF or a #GP(0), because the act of getting to the exception handler may deliver another BTS. **It is highly recommended that the host VMM enables notification TD exit**, as described in 10.12.4.
Processor Trace (PT)	• IA32_RTIT_CONTROL • Requires VMM's consent on TD initialization by setting TD_PARAMS.XFAM[8] to 1	PT state handling as part of the extended feature set state is discussed in 10.5. •
Architectural Last Branch Records (LBRs)	• IA32_LBR_CONTROL • Requires VMM's consent on TD initialization by setting TD_PARAMS.XFAM[15] to 1	LBR state handling as part of the extended feature set state is discussed in 10.5.
Non-Architectural LBRs	• IA32_DEBUGCTL bit 0 (LBR)	Guest TD attempt to set IA32_DEBUGCTL[0] is ignored by the CPU.

14.2. On-TD Performance Monitoring

Intel SDM, Vol. 3, 18 Performance Monitoring

14.2.1. Overview

The host VMM controls whether a guest TD can use the performance monitoring ISA using the TD's ATTRIBUTES.PERFMON bit – part of the TD_PARAMS input to TDH.MNG.INIT (see 20.2.1).

By design, if a guest TD is allowed to use performance monitoring:

- The guest TD enumerates native Perfmon capabilities via CPUID leaf 0x0A.
- The guest TD is allowed to use all Perfmon ISA. This includes CR4.PCE, the RDPMC instruction and the Perfmon MSRs (see 14.2.2 below).
- Perfmon state is context-switched by the Intel TDX module across TD entry and exit transitions.

Context-switching the Perfmon state has a performance impact. TD entry and exit latencies are longer than when a guest TD is not allowed to use Perfmon.

By design, if a guest TD is not allowed to use performance monitoring:

- The guest TD enumerates no Perfmon capabilities. CPUID leaf 0x0A returns all 0s.
- The guest TD is not allowed to use Perfmon ISA.
- Perfmon state is not context-switched across TD entry and exit transitions.

Regardless of Perfmon enabling, per the design:

- IA32_DS_AREA MSR is context-switched across TD entry and exit transitions.
- Counter freeze control (IA32_DEBUGCTL bit 12) is context-switched across TD entry and exit transitions.
- The uncore PMI enable bit (IA32_DEBUGCTL bit 13) is preserved during SEAM mode execution, including Intel TDX module and guest TD execution. This bit is virtualized to the guest TD as 0, and the TD is prevented from setting it. See 14.4 below for details.

See also 14.1 above.

The Intel TDX module is designed to support performance monitoring as implemented on the GLC core:

- Architectural performance monitoring version 5, described in [Intel SDM, Vol. 3, 18.2.5)
- Exactly 8 performance monitoring counters (IA32_PMC0 through IA32_PMC7)
- Exactly 4 fixed counters (IA32_FIXED_CTR0 through IA32_FIXED_CTR3)
- Some non-architectural MSRs (see 14.2.2 below)

14.2.2. Performance Monitoring MSRs

Perfmon uses the following MSRs:

Table 14.2: Performance Monitoring MSRs

MSR	Comments	Enumeration	Reference
IA32_PMCx	multiple MSRs	CPUID(0x0A).EAX[15:8] The Intel TDX module requires the CPU to support 8 counters.	
IA32_PERFEVTSELx	multiple MSRs	CPUID(0x0A).EAX[15:8]	
MSR_OFFCORE_RSPx	2 MSRs, model-specific		
IA32_FIXED_CTRx	multiple MSRs	IA32_FIXED_CTRx is supported if (x < CPUID(0x0A).EDX[4:0]) or if (CPUID(0x0A).ECX[x] == 1). The Intel TDX module requires the CPU to support counters 0 through 3.	[Intel SDM, Vol. 3, 18.2.5.2]
IA32_PERF_METRICS			

MSR	Comments	Enumeration	Reference
IA32_PERF_CAPABILITIES			
IA32_FIXED_CTR_CTRL			
IA32_PERF_GLOBAL_STATUS			
IA32_PERF_GLOBAL_CTRL			
IA32_PERF_GLOBAL_STATUS_RESET			
IA32_PERF_GLOBAL_STATUS_SET			
IA32_PERF_GLOBAL_INUSE			
IA32_PEBS_ENABLE	model-specific		
MSR_PEBS_DATA_CFG	model-specific		
MSR_PEBS_LD_LAT	model-specific		
MSR_PEBS_FRONTEND	model-specific		
IA32_A_PMCx	multiple MSRs	CPUID(0x0A).EAX[15:8], IA32_PERF_CAPABILITIES[13] The Intel TDX module requires the CPU to support 8 counters.	[Intel SDM, Vol. 3, 18.2.6]

MSR virtualization is described in 10.7.

14.2.3. Performance Monitoring Interrupts (PMIs)

By design, when a guest TD is allowed to use Perfmon, it can also configure the counters to raise PMI on overflow. When such a TD counter overflows, the physical interrupt or an NMI configured by the host VMM into the local APIC is delivered. This interrupt or NMI causes a VM exit, and it is delivered as a TD exit to the host VMM. The host VMM is then expected to inject the PMI into the guest TD, either as a virtual interrupt using the posted interrupt mechanism (see 10.9.3), or as virtual NMI using the NMI injection interface (see 10.9.5).

Since the host VMM is not trusted, the guest TD must be ready to handle spurious, delayed or dropped PMIs. Thus, it is recommended for the guest TD to use PEBS instead of PMIs in order to record TD state at counter overflows.

Uncore PMIs are discussed in 14.4 below.

14.3. Off-TD Debug

A guest TD is defined as **debuggable** if its ATTRIBUTES.DEBUG bit is 1. In this mode, the host VMM can use Intel TDX functions to read and modify TD VCPU state and TD private memory, which is not accessible when the TD is non-debuggable.

A debuggable TD is, by nature, untrusted. Since the TD's ATTRIBUTES are included in the TDREPORT_STRUCT, the TD's debuggability state is visible to any third party to which the TD attests.

The applicable Intel TDX functions are listed in Table 14.3 below. Note that some of the functions can access non-secret guest TD state regardless of the DEBUG attribute. The lists of state information that can be read and/or written in non-DEBUG and in DEBUG modes are detailed in the referenced sections.

Table 14.3: Off-TD Debug Interface

Intel TDX Function	ATTRIBUTES.DEBUG = 0	ATTRIBUTES.DEBUG = 1	References
TDH.MNG.RD/ TDH.MNG.WR	N/A	Access secret and non-secret TD-scope state in TDR and TDCS.	21.1, 22.2.22, 22.2.23
TDH.MEM.SEPT.RD	Read Secure EPT entry	Read Secure EPT entry	22.2.12

Intel TDX Function	ATTRIBUTES.DEBUG = 0	ATTRIBUTES.DEBUG = 1	References
TDH.VP.RD/ TDH.VP.WR	Access non-secret TD VCPU state in TDVPS (including TD VMCS)	Access secret and non-secret TD VCPU state in TDVPS (including TD VMCS).	21.2
TDH.MEM.WR/ TDH.MEM.RD	N/A	Access TD-private memory.	22.2.25, 22.2.25
TDH.PHYMEM.PAGE.RDMD	Read page metadata (PAMT information)	Read page metadata (PAMT information).	22.2.28

14.3.1. Modifying Debuggable TD's State, Controls and Memory

When the TD is debuggable, the off-TD debugger can:

- Read and modify TDVMCS fields that contain guest state, VM entry load controls, VM exit save controls, and VM execution controls.
- Read and modify TDVPS fields that contain additional TD VCPU's state (e.g. extended register state).
- Read and modify a per-VCPU copy of the TD's extended feature mask (XFAM), such that more extended register state would be saved to TDVPS on TD exit and restore from TDVPS on TD entry.

This may cause the next VM entry into the TD VCPU to fail due to bad guest state. It may also generate VM exits that wouldn't have happened otherwise (e.g., VM exit due to a #PF within the TD). In non-debuggable TD such VM exits are not expected, and thus treated as fatal TDX module error and lead to shutdown. In debuggable TDs, however, such VM exits are expected and cause TD exit.

Specifically, the TDX module handling of TD VM exits is *extended* as follows:

1. If this TD VM exit might happen on non-debuggable TDs:
    1.1. Do "standard" handling (may result a TD exit).
    1.2. If an exception is pending to be injected into the TD:
        1.2.1. If the TD is debuggable and its exception bitmap is programmed to intercept that exception:
            1.2.1.1. TD exit to the VMM, as if the exception has been raised during TD execution.
    1.3. Resume the TD (may inject an exception).
2. Else (an unexpected VM exit happened):
    2.1. If the TD is debuggable then TD exit.
    2.2. Else handle this as a fatal error.

In any case, the security of other guest TDs running in production mode is not impacted.

14.3.2. Preventing Guest TD Corruption of DRs

The host-side debugger may need to have full control over guest DRs to help prevent their corruption by the guest TD. To do so, the debugger can do the following:

- Use TDH.VP.WR to set the TD VMCS GUEST_DR7 field's Global Detect bit.
- Set the TD VMCS exception bitmap execution control to intercept debug exceptions.

14.4. Uncore Performance Monitoring Interrupts (Uncore PMIs)

By design, neither the Intel TDX module itself not its guest TDs are allowed to use Uncore PMIs. The state of IA32_DEBUGCTL MSR bit 13 (ENABLE_UNCORE_PMI) is preserved across SEAMCALL, SEAM root and non-root mode and SEAMRET, except for very short time periods immediately after SEAMCALL and VM exit.